НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
зі спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Удосконалений спосіб побудови систем управління розумним будинком»

Виконав:
студент VI курсу, групи ТІ-91мп
Олійник Костянтин Володимирович _____

Керівник:
Доцент кафедри ІТМ ІТС, доцент, к.т.н.
Кононова Ірина Віталіївна _____

Рецензент:
Доцент кафедри ТК ІТС, доцент, к.т.н.
Явіся Валерій Сергійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Лариса ГЛОБА

«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Олійнику Костянтину Володимировичу

1. Тема дисертації «Удосконалений спосіб побудови систем управління розумним будинком», науковий керівник дисертації доцент кафедри інформаційно-телекомунікаційних мереж ІТС Кононова Ірина Віталіївна, доцент, к.т.н., затверджені наказом по університету від «03» листопада 2020 р. № 3208-с

2. Термін подання студентом дисертації 10.12.2020 р.

3. Об’єкт дослідження: функціонування системи управління розумним будинком

4. Предмет дослідження: протоколи передачі даних в системі управління розумним будинком

5. Перелік завдань, які потрібно розробити:
- провести аналіз технологій, що використовуються в пристроях розумного будинку
- провести огляд сучасних систем автоматизації в розумних будинках
- визначення та аналіз пріоритетних способів інтеграції пристроїв з системами автоматизації
- вибір апаратного забезпечення та інструментів його програмування
- вдосконалення існуючі архітектури побудови пристроїв розумного дому
- проведення тестування розробленого програмного забезпечення

6. Орієнтовний перелік ілюстративного матеріалу:
   - опис сучасних технології пристроїв розумного дому та сучасних систем автоматизації
   - архітектури побудови пристроїв розумного дому
   - архітектури мікроконтролерів пристроїв розумного будинку
   - порівняльні характеристики протоколів обміну даними
   - опис кодової бази програмного забезпечення
   - опис процесів тестування програмного забезпечення

7. Орієнтовний перелік публікацій

8. Дата видачі завдання 01.09.2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Провести аналіз технологій, що використовуються в пристроях розумного будинку	01.01.2020	виконано
2	Провести огляд сучасних систем автоматизації задач в розумних будинках	01.03.2020	виконано
3	Визначити основні функціональні та нефункціональні вимоги до програмного забезпечення	01.05.2020	виконано
4	Визначити пріоритетні способи інтеграції пристроїв з системами автоматизації	01.06.2020	виконано
5	Обрати та проаналізувати спосіб інтеграції	10.06.2020	виконано
6	Провести аналіз та вибрати необхідне апаратне забезпечення та інструменти його програмування	01.08.2020	виконано
7	Покращити існуючі архітектури побудови пристроїв розумного дому	01.09.2020	виконано
8	Провести тестування розробленого програмного забезпечення	01.10.2020	виконано
9	Розробити стартап-проект	01.11.2020	виконано
10	Оформити магістерську дисертацію	10.12.2020	виконано

Студент Костянтин ОЛІЙНИК

Науковий керівник дисертації Ірина КОНОНОВА

РЕФЕРАТ

Робота містить 141 сторінку, 68 рисунків та 16 таблиць. Було використано 25 джерел.

Актуальність роботи: На сьогоднішній день системи "розумного дому" отримали великого розголосу в сучасному суспільстві, бо вони допомагають заощадити пару важливих ресурсів людського життя - час і гроші. Інноваційні розробки подібного роду спрямовані не тільки на підвищення зручності життя, але і на поліпшення енергозбереження приміщень. Простим прикладом такої автоматизації служить освітлення вашого холодильника. Коли дверцята відкриті - освітлення активно, дверцята зачинені - освітлення вимикається. Таким чином можна виключити неефективне використання електроприладів і опалювальних систем, а якщо врахувати, що ціна на енергоресурси постійно зростає - це дозволить отримати істотну економію як енергії, так і грошових коштів.

Більшість сучасних реалізації програмного забезпечення пристроїв сумісні лише з системи контролю своїх же виробників, що унеможливлює використання пристроїв різних вендорів в рамках однієї інфраструктури. Саме тому універсальне рішення для будь-яких пристроїв на базі єдиного мікроконтролера дозволить використовувати девайси різних виробників та навіть пристрої домашнього виробництва в рамках однієї систему управління домом, що підтримує єдиний протокол з периферією.

Метою роботи є вдосконалення архітектури пристроїв розумного дому та розробка програмного забезпечення для них на основі запропонованої архітектури.

Задачі дослідження:

- провести аналіз технологій, що використовуються в пристроях розумного будинку
- провести огляд сучасних систем автоматизації в розумних будинках

- визначення та аналіз пріоритетних способів інтеграції пристроїв з системами автоматизації
- вибір апаратного забезпечення та інструментів його програмування
- вдосконалення існуючі архітектури побудови пристроїв розумного дому
- проведення тестування розробленого програмного забезпечення
- розробка стартап-проекту, в основі якого буде використовуватися удосконалена архітектура побудови пристроїв розумного будинку

В даній роботі розглядаються сучасні технології, що використовуються для побудови пристроїв розумного будинку. На Основі їх аналізу було обрано оптимальний варіант і проведено аналіз сумісного апаратного забезпечення для побудови пристроїв. На основі проведеного аналізу було обрано необхідні технології, інструменти та протоколи комунікації. Після чого запропоновано варіант удосконаленої архітектури, та розроблено та протестовано програмне забезпечення на реальних пристроях. Доказом удосконалення архітектури стала покращена процедуру завантаження та розробки нового програмного забезпечення та інтеграція з сучасними системами автоматизації задач розумного будинку.

Об’єкт дослідження: функціонування системи управління розумним будинком.

Предмет дослідження: протоколи передачі даних в системі управління розумним будинком.

Наукова новизна: компілятор універсального програмного забезпечення на основі конфігураційного файлу.

Практична значимість: можливість недосвідченим користувачам самостійно реалізовувати проекти розумного дому на базі представленого програмного забезпечення.

Методи дослідження: основними методами дослідження є емпіричні підходи порівняння, експериментування та матеріального моделювання.

Ключові слова: розумний будинок, мікроконтролер, Wi-Fi, MQTT, ESP.

ABSTRACT

The work contains 141 pages, 68 figures and 16 tables. 25 sources were used.

The actuality of the work: Today, "smart home" systems have gained much notoriety in modern society because they help save a couple of important resources of human life - time and money. Innovative developments of this kind are aimed not only at improving the comfort of living, but also at improving energy efficiency. A simple example of such automation is the lighting of your refrigerator. When the door is open - the lighting is active, the door is closed - the lighting is turned off. Thus, it is possible to exclude inefficient use of electrical appliances and heating systems, and given that the price of energy is constantly rising - it will save significant energy and money.

Most modern implementations of device software are compatible only with the control system of their own manufacturers, which makes it impossible to use devices from different vendors within the same infrastructure. That is why a universal solution for any device based on a single microcontroller will allow you to use devices from different manufacturers and even home-made devices within a single home management system that supports a single protocol with peripherals.

The aim of the work is to improve the architecture of smart home devices and develop software for them based on the proposed architecture.

Research objectives:

- to analyze the technologies used in smart home devices
- to review modern automation systems in smart homes
- identification and analysis of priority ways of integration of devices with automation systems
- choice of hardware and tools for its programming
- improving the existing architecture of building smart home devices
- testing of the developed software
- development of a startup project, which will be based on an improved architecture for building smart home devices

This paper considers modern technologies used to build smart home devices. Based on their analysis, the optimal option was selected and the analysis of compatible hardware for building devices was performed. Based on the analysis, the necessary technologies, tools and communication protocols were selected. After that, a variant of the improved architecture was proposed, and software on real devices was developed and tested. Evidence of improved architecture was the improved procedure for downloading and developing new software and integration with modern smart home automation systems.

Object of research: operation of the smart home management system.

Subject of research: data transmission protocols in the smart home management system.

Scientific novelty: compiler of universal software based on a configuration file.

Practical significance: the opportunity for inexperienced users to independently implement a smart home project based on the presented software.

Research methods: the main research methods are empirical approaches to comparison, experimentation and material modeling.

Keywords: Sensor wireless network, wireless network architecture, main node selection algorithms, cluster architecture, method of designing the structure of a wireless sensor network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 10
ВСТУП 11
РОЗДІЛ 1 13
АНАЛІЗ СУЧАСНОГО ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ В ПРИСТРОЯХ РОЗУМНОГО БУДИНКУ 13
1.1 Проводові технології 15
1.2 Безроводові технології 22
1.3 Проводові технології 30
Висновки 32
РОЗДІЛ 2 34
ОГЛЯД СУЧАСНИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ 34
2.1 Види систем управління розумним домом 34
2.2 Системи з відкритим вихідним кодом 35
2.3 Вибір систем для тестування покращеної архітектури 64
Висновки 65
РОЗДІЛ 3 66
ВИБІР ІНСТРУМЕНТІВ І АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УДОСКОНАЛЕННЯ АРХІТЕКТУРИ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ 66
3.1 Визначення основних функціональних та нефункціональних вимог до архітектури програмного забезпечення пристроїв 66
3.2 Визначення способу інтеграції з існуючими системами управління .. 67
3.3 Вибір універсальної самодекларативної конвенції для опису стану девайсу 70
3.4 Апаратне забезпечення для побудови власних пристроїв розумного будинку 81
3.5 Вибір інструментів для створення програмного забезпечення пристроїв розумного будинку 91
Висновки 94

РОЗДІЛ 4 96

ВАРІАНТ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З УДОСКОНАЛЕНОЮ АРХІТЕКТУРОЮ 96

4.1 Реалізація автоматичної збірки програмного забезпечення на основі конфігураційного файлу 96

4.2 Створення модульної бази для роботи с сенсорами та керованими пристроями та розробка інструментів для програмного забезпечення ... 102

4.3 Створення базового інтерфейсу для підключення пристроїв в локальну мережу 110

4.4 Приклад тестування програмного забезпечення 113

Висновки 120

РОЗДІЛ 5 122

РОЗРОБКА СТАРТАП-ПРОЕКТУ 122

5.1 Опис ідеї проекту 122

5.2 Технологічний аудит ідеї проекту 124

5.3 Аналіз ринкових можливостей запуску стартап проекту 124

5.4 Розроблення ринкової стратегії проекту 130

5.5 Розроблення маркетингової програми стартап-проекту 132

Висновки 132

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ **134**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **136**

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПК	Персональний комп'ютер
ПЗ	Програмне Забезпечення
AMQP	Advanced Message Queuing Protocol
CoAP	Contained Application Protocol
IoT	Internet of Things
HTTP	Hyper Text Transport Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
OSI	Open Systems Interconnection
QoS	Quality of Service
REST	Representational State Transfer
RF	Radio frequency
SSL	Security Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
USB	Universal Serial Bus
UDP	User Datagram Protocol
XML	eXtensible Markup Language

ВСТУП

На сьогоднішній день сфера автоматизації задач в системах контролю розумним будинком стає все більш і більш актуальною. Все більше людей встановлюють вдома системи розумного будинку для вирішення повсякденних задач, економії ресурсів та налаштування безпеки власного оселі. В зв'язку з цим ринок розробки розумних пристроїв розвивається значними темпами. Інженери робочих груп впроваджують нові протоколи, покращують існуючі і роблять це з увагою до IoT.

Але не дивлячись не величезний попит недосвідченим користувачам, які хочуть отримувати готові рішення своїх задач, важко знайти повноцінні рішення від одних виробників, що змушує їх використовувати девайси різних брендів, а отже і різне програмне забезпечення для їх керування, що значно погіршує користувальницький досвід і змушує встановлювати безліч утиліт на персональні комп’ютери та смартфони.

Саме для вирішення такої проблеми на ринку не так давно почали з’являтись уніфіковані глобальні системи розумного будинку, які дозволяють приєднати багато девайсів в єдину екосистему і керувати ними з одного місця.

В зв’язку з появою таких систем виробники апаратної частини змушені підлаштовуватись під їх вимоги для можливих інтеграцій, але ця задача досі не реалізовано в більшості пристроїв. Це змушує користувачів самостійно розбиратись в процесах інтеграції або робити запити до розробників платформ для реалізації необхідного функціоналу на їх стороні.

В роботі наведено пропозиції по покращенню архітектури побудови пристроїв розумного будинку за рахунок абстрагування від систем управління і винесення процесів інтеграції на інструменти платформ за рахунок використання стандартних протоколів та підходів до обміну даними. За рахунок використання стандартизованих методів та конвенцій, підтримка яких є у більшості система, запропонована архітектура дає можливість інтегрувати пристрої в будь-які системи за мінімальний час.

Крім цього в удосконаленій архітектурі запропоновано метод збору програмного забезпечення на основі файлів конфігурації, який дозволить простим користувачам самостійно розробляти та макетувати свої пристрої на основі простих та популярні комплектаційних матеріалів, що значно знижує собівартість пристроїв і дає можливість користувачам конфігурувати їх самостійно з акцентом на ті задачі, які необхідно вирішувати в першу чергу.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ В ПРИСТРОЯХ РОЗУМНОГО БУДИНКУ

На сьогоднішній день питання домашньої автоматизації стало дуже популярним терміном в усьому світі. Кожен користувач може легко керувати побутовою технікою, освітленням, опаленням та охолодженням, електричними розетками і навіть мультимедіа або безпекою за допомогою мережевих пристрою, таких як ПК, планшет або смартфон. Крім того, можна контролювати і виконувати енергоменеджмент. Будь-хто може додати в свій будинок датчики або інтелектуальні пристрої без необхідності вносити будь-які зміни в існуючу установку або обладнання. Він може віддалено перевірити свій домашній статус і виконати такі дії, як ввімкнення чи вимкнення приладу (електричний праска) або підігрівати до тепла воду перед його приходом.

Ринок розумного будинку став доступним і досяжним з появою безлічі компаній і протоколів щороку. Всесвітньо відомі комп'ютерні компанії, такі як Apple (HomeKit), Samsung (SmartThings) і Google (Alexa) вклали кошти в цей ринок, зробивши його доступним для домашніх користувачів. Є також такі компанії, як Siemens (KNX) і Dupline, які розробляють рішення багато років.

У зв'язку з поширенням великої кількості датчиків в секторі додатків домашньої автоматизації та впровадженням технології IoT (Інтернет речей) виникла необхідність в домашній мережі, щоб використовувати їх переваги. Тому була створена модель, показана на рисунку (рис.1.1).

На сучасному ринку домашньої автоматизації існує безліч варіантів установки: провідний, бездротової або обидва та багато інших.

Популярні протоколи і мережеві комунікаційні технології, починаються з відомих, таких як Bluetooth, Wi-Fi, GSM, ZigBee, ZWave і KNX, до менш відомих — EnOcean, Insteon і новими багатообіцяючими, такими як LowPAN, Thread і DASH, LoRA або Sigfox.

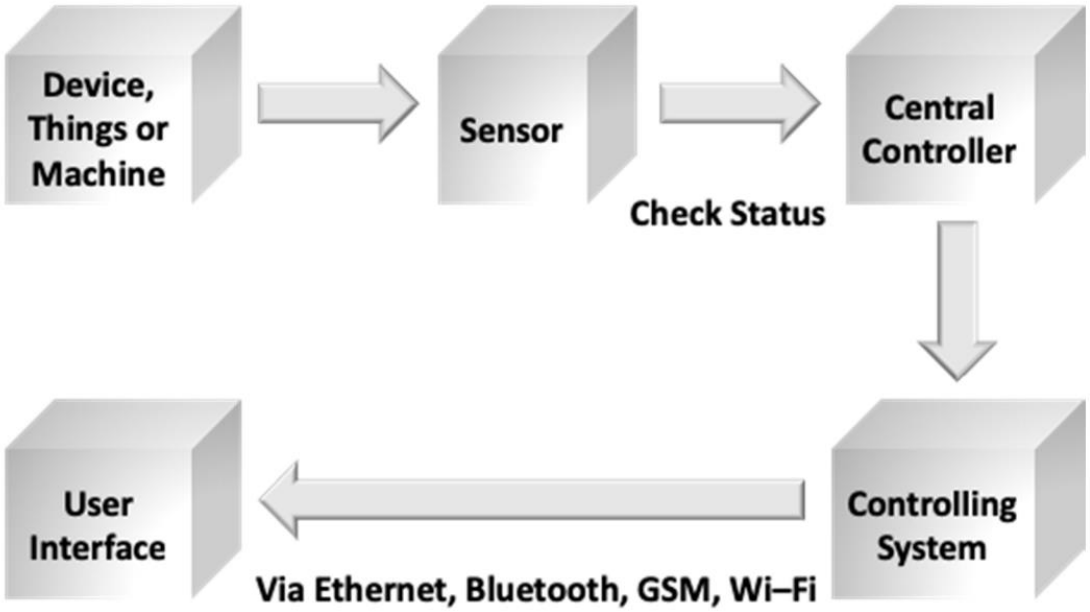


Рис. 1.1 Internet of Things технології для домашньої автоматизації

На рисунку представлено IoT технології для домашньої автоматизації. Основними елементами є Things or Machine - девайс чи пристрій, Sensor – датчик, Check status - перевірка статусу, Central Controller - центральный контроллер, Controlling System -система контролю, протокол передачі данних Via Ethernet, Bluetooth,GSM,Wi-Fi та інтерфейс користувача User Interface.

Різноманітність варіантів може збити з пантелику користувачів і ускладнити правильний вибір, оскільки кожна з цих технологій має деякі переваги і недоліки з різними рівнями технологічної зрілості. Іноді в одному будинку співіснують дві або більше з цих технологій, змушуючи користувача використовувати різні контролери або додатки для управління ними [1].

1.1 Проводові технології

Щоб пристрої могли обмінюватися даними, необхідно визначити протокол, що забезпечує стандарти, політики, процедури та формати для обміну даними. Багато протоколів створені виключно для мереж домашньої автоматизації, в той час як інші беруть початок в уже існуючих протоколах, які використовуються в комунікаціях і мережах (Wi-Fi, Bluetooth, GSM). Крім того, є ті, що використовуються в IoT (Інтернеті речей) і можуть використовуватися в системі домашньої автоматизації. Щороку видається новий протокол або варіант, в той час як є інші, які добре зарекомендували себе протягом багатьох років, з часом об'єднують безліч поліпшень. У наступних розділах дається короткий опис найбільш популярних протоколів.

BACnet

Це протокол для автоматизації будівель і мереж управління. Вперше він був розроблений комітетом, сформованим Американським товариством інженерів з опалення, охолодження та кондиціювання повітря (ASHRAE, метою якого є розробка інтероперабельних систем домашньої автоматизації. Його специфікація складається з трьох основних частин:

1. Подання обладнання автоматизації будівель в стандартному вигляді.

2. Визначення форматів повідомлень, які можуть бути відправлені в комп'ютерній мережі для моніторингу.

3. Контроль обладнання і визначення того, на яких LAN можуть використовуватися для зв'язку BACnet.

Всього існує 35 типів повідомлень (сервісів), розділених на п'ять класів. Це доступ і управління властивостями об'єктів, сигналізації і події, завантаження і скачування файлів, управління віддаленими пристроями, функції віртуального терміналу [2].

LonWorks

Local Operation Networks (LonWorks) - це платформа, розроблена компанією Echelon Corporation в 1991 році для підтримки керуючого додатку.

Кожен з пристроїв включає в себе нейронний чіп, трансивер і прикладну електроніку. Перший компонент складається з системи на кристалі, що включає кілька мікропроцесорів, RAM, ROM і порти інтерфейсу введення-виведення. Другий компонент - це електронний модуль, що забезпечує фізичний інтерфейс для зв'язку з іншими пристроями.

Метою даної технології є поділ пристроїв або систем на групи інтелектуальних елементів (вузлів), пов'язаних між собою засобами передачі даних (кручена пара, Ethernet), створюючи розумну мережа. Цей протокол використовує модель OSI, надаючи послуги на всіх семи рівнях. Це як складний, так і закінчений протокол, що забезпечує функціональні можливості для кожного рівня. На жаль, для забезпечення функціональності своїх систем постачальники повинні порушити підписаний контракт, щоб приєднатися до LonWorks, що позбавляє їх права використовувати знак платформ або переконувати золотого учасника схвалити їх розширення.

Інша проблема полягає в тому, що не всі системи належать LonMark, глобальної членської організації, створеної для просування і просування бізнесу щодо ефективної та дієвої інтеграції відкритих систем управління від різних постачальників, що використовують ISO / IEC 14908-1 і пов'язані стандарти. LonWorks, ймовірно, буде використовуватися деякими постачальниками, які не входять до консорціуму платформи [3].

KONNEX (KNX)

Це відкритий стандарт, застосовуваний в будинках і будівлях. Він використовує переваги майже всього, що використовується в домашніх комунікаціях, таких як кручена пара (TP), лінія електропередачі (PL), радіочастота (RF) і Ethernet, технологія, яка використовує Інтернет-протокол (IP). KNX не тільки об'єднує, а й розширює три застарілих стандарту шини: EIB, EHS і BatiBUS.

KNX може з'єднувати різнорідні шинні пристрою незалежно від використовуваного середовища (TP, PL, RF, IP) на своєму фізичному рівні для

обміну інформацією в заздалегідь визначеному стандарті. Кожен пристрій може діяти як інтелектуальний датчик, який визначає ситуації навколишнього середовища, такі як погодні умови, умови освітлення, системи моніторингу, присутність людей і інтелектуальний виконавчий механізм, керуючий обладнанням будівлі (мультимедійні пристрої, жалюзі, віконниці, пристрої безпеки, енергоменеджер, опалення, вентиляція, кондиціонер). Кожен пристрій може сигналізувати, контролювати і контролюватися з використанням загального протоколу без необхідності використання додаткових центрів управління.

Цей стандарт зв'язку є еталонном моделі OSI, за винятком об'єднання трьох верхніх рівнів в один. В результаті базових п’яти рівнів: фізичний, канал передачі даних, мережа, транспорт і додаток.

Конфігурація KNX не є технологією plug and play, як інші протоколи, і для правильної роботи підмережі, її необхідно налаштовувати вручну з використанням програмного забезпечення. Для зберігання та обміну конфігураціями з іншими пристроями в мережі він використовує формат даних XML.

Платформа в її стандартній формі заснована на концепції точок даних, входів і виходів, а її параметри представляють собою екземпляри точок даних з контейнерів, груп або властивостей об'єктів. Точки даних стандартизируются, а потім групуються в функціональні блоки, при цьому їм присвоюються ім'я, тип і формат даних. Залежно від сценарію випадку KNX працює в 4 робочих режимах: системний режим, режим конфігурації, режим кнопки і режим логічних тегів.

Що стосується топології, то вона безкоштовна і децентралізована з розподіленим інтелектом, який може виявляти, контролювати і відслідковувати послідовності і функції за єдиною лінії зв'язку. Архітектура, оскільки вона здатна до масштабування, дозволяє розширювати систему, щоб покрити потреби будівлі, щоб забезпечити наскрізну зв'язок для датчиків і

виконавчих механізмів з малим часом відгуку в ситуаціях динамічного управління.

Для того, щоб KNX міг забезпечити інтеграцію до стандартів Інтернету, в рамках ініціативи був розроблений відкритий стандарт - розширена мережева топологія для уніфікованої інтеграції будівель (ANubis). Його основна мета - економічне впровадження, починаючи з малих і закінчуючи великими будинками, з використанням ієрархічних шин.

Завдяки використанню всіх згаданих характеристик, KNX підходить для використання в домашньої автоматизації. Використовувана комерційна лінія називалася Instabus, яка була розроблена Siemens і сертифікована EIBA, приділяючи особливу увагу споживанню енергії в житлових приміщеннях, пропонуючи такі рішення, як управління освітленням, HVAC (тепло, вентиляція і повітря кондиціювання) управління і контроль безпеки. Її можна охарактеризувати як розподілену систему без використання центрального контролера, щоб користувач міг конфігурувати, програмувати і управляти пристроями системи за допомогою комп'ютера, підключеного до послідовного інтерфейсу RS 232, з використанням програмного забезпечення EIB [4].

В останні роки автори Ruta, Di Sciascio, Scioscia і Loseto опублікували статті для KNX, що пропонують:

1. Розвиток протоколу KNX в семантиці, здатне узгоджувати потреби користувачів, виражені в їх анотованих профілях, з використанням повністю автоматизованої інфраструктури побутових пристроїв, заснованої на онтології.

2. Впровадження семантичного розширення на рівні додатків за стандартом KNX, зі збереженням успадкованих додатків і відкриттям можливостей для подальших поліпшень.

3. Підхід, заснований на семантиці, здатний взаємодіяти з користувачами і пристроями в домашніх інфраструктурі, характеристики якої виражаються за допомогою анотованих профілів.

Ethernet

Протокол Ethernet - одна з найпопулярніших мережевих технологій і набула своєї популярності за рахунок використання в сучасних комп'ютерних

мережах і комунікаціях з використанням VoIP (voice over IP - голос через IP). Крім того, він широко використовується в автоматизації будинків, інтернеті речей та навіть у мобільному зв'язку і автомобілебудуванні. Ethernet, за словами його винахідника (Боба Меткалф) характеризується як інноваційна група і має сім основних атрибутів:

- нативний режим підключення до Інтернету
- висока швидкість
- мультимедійний
- стандартизованість (IEEE 802.3)
- відсутність відкритого коду
- забезпечення сумісності (використання plug-and-play) та зворотня сумісність

Цей стандарт використовує протокол TCP / IP, і відповідно до його 6-ї версії забезпечує можливість підключення 2^32 пристроїв. Крім того, він може підтримувати передачу енергії, забезпечуючи можливість підключеним пристроям використовувати одне і те ж середовище (кабель, виту пару) як своє джерело живлення. Згідно з мережевою моделлю OSI, Ethernet, як і KNX, об'єднав прикладний рівень, рівень представлення та сеансовий в один прикладний шар.

Для провідного підключення пристроїв використовується вита пара або оптоволокно. Перший режим підключення може підтримувати передачу даних і живлення. Швидкість передачі даних за допомогою витої пари при використанні протоколу Ethernet вже досягає 400 Гбіт / с . Існує два режими передачі по оптоволокну: одномодовий, і багатомодовий. Очікується що при першому режимі швидкість передачі даних досягне 400 Гбіт / с на відстань до

40 км., в той час як в багатомодовому режимі очікується швидкість передачі даних до 1,6 Тбіт / с.

Крім кінцевих точок доступу в даній мережі використовуються такі пристрої як комутатори, шлюзи і маршрутизатори. Комутатори використовуються для підключення пристроїв в безпечному режимі в локальних мереж. Шлюзи використовуються для підключення пристроїв які знаходяться в різних мережах або для з'єднання різних локальних мереж. Маршрутизатори працюють як шлюзи, але також забезпечують підключення інших пристроїв і хостів через хмару.

X10

Цей протокол використовується для дистанційного керування передавачами / приймачами відповідності X-10 через лінію електропередач та електричну проводку яка прокладена в будинку. Вперше він був представлений такими компаніями як Radio Shack, системою управління Sears, Power System and Power Plug, а зараз використовується й іншими компаніями.

Команди з спеціальний унікальними ідентифікаторами транслюються передавачами до приймального блоку, з використанням точок доступу, розеток та powerline адаптерів, розташованих у будинку. За допомогою унікального ідентифікатора повідомлення зчитуються одержувачами які визначають чи даний ідентифікатор відповідає його власному ідентифікатору, і у випадку позитивного результату він продовжує читати ціле повідомлення, щоб обробити команду.

Стандарт визначає 256 адресів, 16 юніт кодів, кожен з яких складається з 16 секторів. У випадках зв'язку в зонах, які перевищують 185 квадратних метрів, можуть виникнути проблеми з передачею, тому потрібен міст або підсилювач сигналу. Крім того, стандарт визначає, як біт повідомлення передається по лінії електропередач із використанням сигналу змінного струм як носія. Заголовок повідомлення починається з 4-значного стартового коду, наступні 4 для коду дому , щоб вказати сектор - задається за допомогою

латинських літер від A до P, а решта 5 цифр - ключовий код, який вказує на одержувача повідомлення. X10 в основному визначається на фізичному рівні, але також має компоненти з рівня каналу передачі даних. Оскільки верхній рівень не є чітко визначеним, і його можна охарактеризувати як такий, де виконуються команди.

Через низьку швидкість носії мають обмеження в мережі, звідки і виникає низька швидкість передачі даних. Через це протокол використовується для забезпечення освітлення, в мережах приладів та мережах датчиків безпеки.

Загальна мова додатків (CAL) - це ще один підхід, розроблений CEBus як універсальна мова спілкування для електричних побутових пристроїв, яка інтерпретується всіма пристроями в будинку, і також використовує для комунікації мережі живлення. Вона була прийнята стандартом EIA 721 і вказується як довідковий посібник в компаніях Home Plug and Play. Потім його було розширено і створено набір правил та інструментів, що забезпечують зв'язок між декількома пристроями з визначенням інтерфейсів для різних фізичних рівнів та протоколів.

Insteon

Insteon - це система, розроблена для того, щоб замінити стандарт X10 за допомогою використання бездротової інтеграції систем електромереж. Його конструкція дозволяє пристроям (датчикам, перемикачам) співпрацювати незалежно від того, підключені вони до ліній електропередач або радіочастот. Це одна з небагатьох технологій, що дозволяє здійснювати комунікації як дротовими, так і бездротовим шляхом.

Вона сумісна з усіма пристроями, що підтримують стандарт X10, хоча їх команди не дуже схожі. Це вдалось забезпечити за допомогою набору мікросхем Insteon, який має можливість відповідати на повідомлення X10, а отже, взаємодіяти з цими пристроями.

Передача даних здійснюється на частотах 132 кГц при провідному підключенні до електропередач та 915 МГц для бездротових. Він підтримує подвійну (повну) сітчасту топологію для обміну повідомленнями від дротової до бездротової. Пристрої можуть підключатись автоматично через відправлення універсальних повідомлень, що зчитуються в обох мережах. За допомогою цього система забезпечує якість сервісу, та виявлення та виправлення помилок.

Щоб забезпечити взаємодію з іншими платформами, як і X10 в Insteon, було створено альянс, який включає виробників цієї платформи, розробників, організації та деякі компанії Fortune 500.

Щодо мережевої моделі OSI, Insteon має ту саму логіку, що і KNX та стандарт Ethernet, і використовує 5 шарів замість 7.

Існує два типи повідомлень, якими обмінюються пристрої в Insteon: стандартне (10 байт) і розширене (24 байти). Перші три байти в пакеті даних описують адресу джерела, а наступні три - адресу призначення, сьомий байт описується як прапор (Тип та інформація про повідомлення), наступні два описують надіслану команду, і останній використовується для перевірки цілісності повідомлення. Також існують розширені повідомлення повідомлення в яких 14 додаткових байтіві передають дані користувача. Insteon може підтримувати велику кількість пристроїв, оскільки підтримує 16777216 унікальних ідентифікатори та 65536 команди [5].

1.2 Безроводові технології

Bluetooth

Перевагою цієї технології персональної мережі для домашньої автоматизації є те що технології Bluetooth, вбудована в кожен смартфон. Смартфони можуть виступати як клієнтська сторона реалізації, тоді як серверна вимагає використання мікроконтролера, який зазвичай інтегрований у домашній шлюз (HAN - Home Area Network - шлюз домашньої мережі). Зв'язок між клієнтом і сервером здійснюється за допомогою протоколу

Bluetooth який забезпечує бездротовий обмін даними з додатком автоматизації. Що стосується безпеки, застосовується захист паролем, щоб будь-який аутентифікований користувач мав доступ до пристроїв.

З введенням стандарту Bluetooth 4.0 37 було впроваджено нову реалізацію, BLE (Bluetooth Low Energy). Впровадження BLE дозволило використання багатьох програм для домашньої автоматизації, оскільки забезпечує комунікацію з низьким енергоспоживанням без втрат эфективносты. Завдяки енергоефективним сценаріям бездротового зв'язку він був прийнятий багатьма розробниками для IoT(інтернет речей), оскільки він використовував переваги існуючих продуктів BLE без необхідності придбання шлюзу. Однак Bluetooth Smart, як його також називають, підтримував лише однорангові та зірочні топології зв'язку. Однак протоколи конкурентів також містять додаткові топології мережі, що спровокувало необхідність введення такої ж можливості, що призвело до розробки специфікації Mesh Network

Як вже згадувалося раніше, BLE як варіант Bluetooth працює в діапазоні ISM 2,4 ГГц і використовує діапазон частот від 2402 МГц до 2480 МГц. Спектр який тут використовується розділений на кілька каналів зв'язку, але на відміну від технології Bluetooth BLE використовує 2 МГц простір між ними, що дозволяє забезпечити 40 каналів (рис.1.2). Три з них використовуються як основні канали, щоб уникнути перешкод з мережею Wi-Fi, які працюють на цій частоті, залишаючи 37 каналів, орієнтованих на з'єднання.

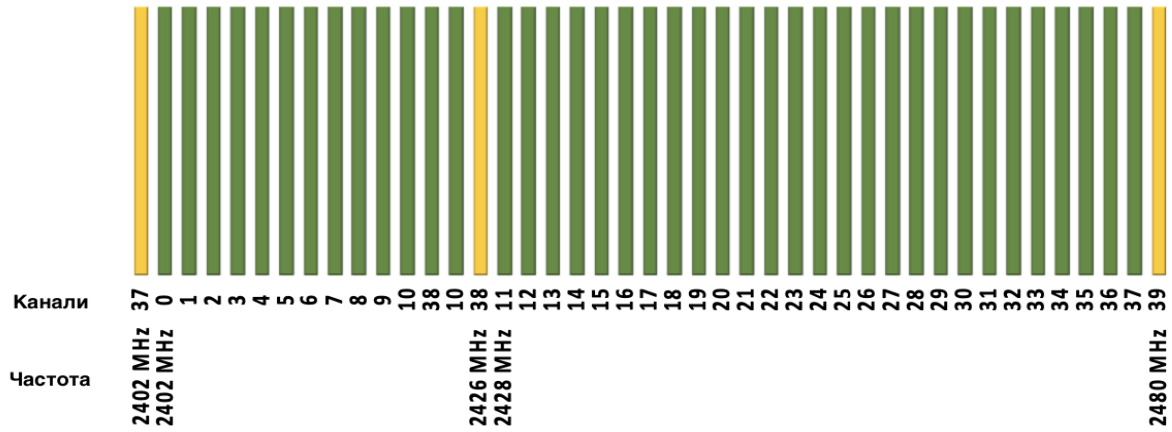


Рис. 1.2. Розподіл каналів у зв'язку BLE

Зв'язок між пристроями може мати два режими: оповіщувальний або орієнтований на з'єднання. Перший - це коли пристрій розсилає інформацію через один із каналів сповіщення з певною роллю. У другому режимі решта 37 каналів використовуються для синхронізації пристроїв, де один з них виконує функції головного управління з'єднаннями з іншими пристроями, які виконують роль ведених пристроїв.

Шари Bluetooth будуються без суворого дотримання структури мережевої моделі OSI. Другий рівень цього стандарту (Baseband) включає частину перших двох рівнів OSI. Третій рівень Bluetooth охоплює деякі функції другого та третього рівнів OSI. Четвертий і останній рівень протоколу - це прикладний та рівень представлення, де всі функції чотирьох верхніх шарів OSI об’єднані в один [6].

Wi-Fi

Хоч ця мережева технологія виглядає досить схожою з Bluetooth, оскільки обидва вони працюють на частоті 2,4 ГГц, вона має багато відмінностей. Крім того, Wi-Fi можна охарактеризувати як бездротовий Ethernet, оскільки обидва вони використовують протокол TCP / IP для зв'язку та мають однакову структуру OSI. Їх єдиною відмінністю є фізичний шар, де середовищем у цій технології є повітря, а не мідна або волоконна оптика.

Ця технологія використовує переваги існуючого домашнього підключення до Інтернету, зокрема локальне бездротове з’єднання. Wi-Fi можна використовувати з будь-якими мережевими пристроями, оскільки зазвичай більшість із них підтримують протокол TCP / IP. Таким чином, із використанням нової версії цього протоколу (IPv6), кожен пристрій може мати свою унікальну адресу IP у мережі. Крім того, він може співпрацювати майже з усіма іншими технологіями автоматизації будинку, щоб забезпечити зв’язок, гнучкість та мобільність для кінцевого користувача. Це досягається за допомогою взаємозв'язку системи автоматизації з користувальницьким

інтерфейсом компьютера, ноутбука або мобільного пристрої - смартофона або планшета.

Основними модулями використання цієї технології є сервер, модуль апаратного інтерфейсу та програмний пакет. Серверні модулі використовують технологію Wi-Fi для зв'язку з локальною мережею, тоді як модулі апаратного інтерфейсу підключені або за спільним протоколом TCP / IP, або за певним власним протоколом. Веб-інтерфейс сервера може бути доступним для локальних або віддалених користувачів за допомогою будь-якого можливого веб-браузера. Програмне забезпечення, що з'єднує сервер з модулями, складається з двох частин: серверної програми та прошивки мікроконтролера.

Деякі технології, що базуються на протоколі TCP / IP, реалізовані в перших трьох рівнях OSI і забезпечують певну функціональність, особливо для мультимедійних пристроїв. Між пристроями забезпечується прямий зв’язок та висока швидкість передачі даних.

Universal Plug and Play (UPnP) технологія застосовується на прикладному рівні, підтримуючи стек протоколів TCP / IP у шарах нижче. Його мета - дозволити пристроям безперешкодно з'єднуватися між собою, одночасно інтегруючи різні виробничі пристрої. Її технологія заснована на використанні веб сервісів, так таких протоколів як IP, протокол керування передачею (TCP), протокол користувальницьких датаграм (UDP), протокол передачі гіпертексту (HTTP), протокол об’єктів простого доступу (SOAP), а також використання розширюваної мови розмітки (XML) та архітектури повідомлень про загальні події (GENA). Як ціль він повинен забезпечити прозору мережу з нульовою конфігурацією та автоматичну процедуру виявлення пристроїв.

У цій архітектурі пристрої мають дві ролі: керовані пристрої(КП) , які є мережевими вузлами, що надають послуги, або контрольні точки(КТ)- це контролери, що виявляють і управляють пристроями. У деяких випадках пристрій може діяти як керований пристрій, так і як контрольна точка. Коли

керований пристрій додається до мережі, він отримує свою IP-адресу, а потім передає свої послуги на контрольну точку за допомогою пакету багатоадресної передачі, що надається Простим протоколом виявлення послуг (SSDP). Якщо до мережі додається пристрій- контрольна точка, він також отримує IP-адресу, а потім виконує пошук необхідних пристроїв в мережі.

Взаємодія з протоколом базується на шести кроках:

- Адресація - коли пристрій зазвичай автоматично отримує свою IP-адресу,
- Discovery - виявлення пристрою з використанням SSDP,
- Опис - описуються сервіси пристроїв із використанням стандартизованого пристрою або послуги UPnP або шаблонних пристроїв,
- Керування - управління пристроями може здійснюватися з використанням трьох механізмів, що провокують дію: дистанційний процедурний виклик (RPC), контрольне повідомлення (CM), XML SOAP із запитами та відповідями по HTTP,
- Підписка - це механізм, що описує підписку контрольної точки на сервіс для надсилання сповіщення коли певні сервісні змінні змінюються,
- Презентація - це веб-інтерфейс, доступ до якого здійснюється веб-браузером, де кінцевий користувач може легко та безпосередньо керувати пристроєм.

Взаємодія починається, коли керований пристрій виявляє контрольну точку, і також отримує опис сервісу з URL-адреси пристрою. Якщо сервіси відповідають вимогам контрольної точки, він робить підписку на них або, залежно від природи сервісу, використовує службу для управління нею. Тоді користувач має можливість отримати доступ до веб-інтерфейсу пристрою, щоб отримати доступ до його опису та елементів керування.

Для того, щоб цей сервіс працюв, домашній маршрутизатор повинен підтримувати та вмикати послугу UPnP. Немає потреби в будь якому іншому

контролері, маршрутизатор діє як сховище, відображаючи кожний сервіс на пристрої, що її надає.

Коли було визначено цю архітектуру, форум UPnP включили у стандартні протоколи керування пристроями (DCP), забезпечуючи способи зв'язку та взаємодії електронних пристроїв. На даний момент було розроблено DCP для аудіо- та відеосистем, принтерів, камер, систем доступу, інтернет-шлюзів та автоматизації будинків. Крім того, остання версія архітектури представляє протокол, який вирішує проблему інтегрованих UPnP і Служби іменування об’єктів (ONS). Це глобальний каталог сервісів, заснований на унікальній ідентифікації ключів EPC та GS1. Відповідно до цього протоколу контрольна точка збирає ідентифікатори активованих пристроїв, а сервіси, пов’язані з ними, можна автоматично отримати від ONS.

DLNA стандарт розглядається як проміжний рівень протоколу UPnP, відповідно до мережового фреймворку AV(Audio Visual). *DLNA* - це торгова організація, сформована Sony, Microsoft та Intel у 2003 році, для визначення вказівок щодо сумісності, що дозволяють обмінюватися цифровими мультимедіа між мультимедійними пристроями. Назва утворена від Digital Living Network Alliance. Його мета полягає у взаємозв'язку дротових та бездротових мультимедійних пристроїв, щоб забезпечити можливість спільного використання та збільшення використовуваних цифрових медіа.

Відповідно достандарту UPnP, архітектура AV прийнята для управління носіями та для управління пристроями. З цієї причини ця архітектура використовується для управління, виявлення та управління носіями на розподілених пристроях. Відповідно достандарту існують два типи пристроїв:

- Сервери цифрових медіа, які відповідають за отримання, запис, зберігання та спільний доступ до мультимедійного вмісту та
- Цифрові медіаплеєри, які здійснюють пошук в мережі для відтворення медіавмісту.

Пристрої, які реалізують внутрішній обмін мережевими медіаресурсами з мережею через мережу DLNA, називаються адаптерами цифрових медіа. Він отримує аудіо- та медіа-файли на комп’ютерах, пристроях чи серверах у мережі, декодує їх у форматі DMA, а потім відображає на іншому пристрої. Для того, щоб запрацював механізм ефективності, заснований на DLNA, потрібно застосовувати різнорідне обладнання, програмне забезпечення, протоколи зв'язку та формати даних. Хоч ця архітектура забезпечує взаємодію користувальницьких додатків із широким спектром даних, пристроїв та служб у мережі, з використанням абстрактних інтерфейсів високого рівня, що мають стандарти веб-служб, тут не враховується економія енергії. Тому було запропоновано механізм енергоефективності для досягнення балансу між забезпеченням якості обслуговування та споживанням енергії. Мета використання данного стандарту - зменшити кількість непотрібних багатоадресних повідомлень [7].

Zigbee

Стандарт бездротових мереж Zigbee широко використовується для впровадження систем домашньої автоматизації. Він забезпечує низьку швидкість передачі даних для обміну даними та є енергоефективним протоколом, який складається з мережевих координаторів, маршрутизаторів та кінцевих пристроїв.

Через затримки в стандартного з’єднанні Bluetooth група компаній розпочала розробку бездротової персональної мережі (WPAN). Цю технологію назвали ZigBee, і її основною метою було недороге та енергоефективне споживання енергії. Стандарт складається з чотирьох шарів: фізичного, рівня контролю доступу, мережевого та прикладного . Два перші шари визначені стандартом 802.15.4, який визначає фізичні шари, які можна використовувати, та способи їх досягнення, інші ж шари визначені протоколом ZigBee. Прикладний рівень має 2 підрівні, де верхній призначений для загальнодоступного профілю, що надає постачальникам можливість

створювати сумісні продукти. Стандарт фізичного радіозв'язку працює в усьому світі в неліцензійних смугах на частотах 2,4 ГГц (глобальний), 915 МГц (Америка) та 868 МГц (Європа). Його швидкість передачі даних може забезпечувати норми в 250 Кбіт / с (16 каналів, 2,4 ГГц), 40 Кбіт / с (10 каналів, 915 МГц) і 20 Кбіт / с (1 канал, 868 МГц).

Координатори мережі реєструють і зберігають продуктивність пристроїв і відповідають за підтримку мережі. Маршрутизатори, використовуючи дані координатора, забезпечують зв'язок між декількома пристроями або кінцевими точками (датчиками) у мережі. Альтернативно для обміну даними використовується мережа Wi-Fi, яка використовує переваги існуючої домашньої мережі та широкої мережевої адресації, що передбачена протоколом IPv6. Це також забезпечує доступ до мережевих пристроїв, таких як ноутбуки, комп'ютери, планшети, смартфони для управління пристроями через домашній шлюз. Ця функція підвищує гнучкість та мобільність. Тому для підвищення безпеки, повідомлення спочатку обробляються основним вбудованим віртуальним алгоритмом. Якщо вони позначені як безпечні, вони будуть зашифровані та направлені на справжній мережевий пристрій у мережі. Використовується в багатьох прикладних сферах таких як керування освітленням, забезпечення безпеки та керування енергоефективністю що полегшує розробку приладів з різних сфер які повинні бути сумісні між собою [8].

Z-wave

Це реалізація, яка складається з мереж, протоколів прикладного рівня та чітко визначеного спілкування, і це забезпечує основу для реалізації Інтернету речей. Z-wave - це недорога бездротова технологія, що дозволяє підключати необхідні девайси до мережі.

Z-wave складається з чотирьох шарів: фізичного, контроля доступу, мережевого та прикладного, який також включає в себе додатковий шар інтерфейсу додатків. Специфікація для перших двох шарів легко доступна у

відкритому доступі, для мережевого ж представлено всього лиш декілька деталей. В реалізації використовується централізовані таблиці маршрутизації, щоб розрахувати маршрути та записати їх у повідомлення, і вказати необхідну поведінку при переадресації. Поки

Fuller описує основний механізм переадресації вихідного маршруту Z-wave, там не так багато інформації про протокол маршрутизації. Отже, реалізація цього протоколу маршрутизації з відкритим кодом не існує. Існує альтернатива OpenZwave з відкритим кодом для ПК, але логіка маршрутизації базується на мікропрограмі USB-трансивера Z-Wave.

Для того, щоб підключити пристрої Z-Wave, кожен з них ідентифікується 4-байтовим основним ідентифікатором, який призначається контролером після процесу створення пари. Ця інформація однакова для всіх підключених вузлів і є унікальною, оскільки контролеру цю інформацію визначає його вендор. Другий ідентифікатор в мережі - це ідентифікатор вузла - це значення одного байта, присвоєного пристрою контролером у процесі сполучення, і воно завжди більше від того, де воно було згенеровано. Контролер завжди має ідентифікатор 1, а пристрої відповідно до порядку їх сполучення мають ідентифікатори 2, 3. Якщо до вузла підключено багато пристроїф їх ідентифікатори збільшується і це може спричинити відмову в використання пристроїв, які мають великий показник ідентифікатору.

Прикладний рівень в даній реалізації визначається досить чітко, щоб мати сумісність з різними контролерами, датчиками, виконавчими механізмами на різних виробництвах. Кожен пристрій не повинен брати участь у кожній транзакції на цьому рівні. Контролеру світла не потрібно знати температурний статус [9].

1.3 Проводові технології

Після ознайомлення з основними технологіями, що використовуються в процесах побудови пристроїв розумного будинки стало зрозумілим, що вони

розвиваються з великими темпами, оскільки кожна з технологій продовжує покращуватись і крім цього регулярно з’являються нові.

Для вибору технології, яка стане основою покращеною архітектури пристрою розумного дому було проведено порівняльний аналіз опрацьованих по декільком критеріям:

- доступність апаратного забезпечення, що підтримує дану технологію
- стандартизація протоколів передачі даних на всіх рівнях роботи
- технічна можливість реалізацію підтримки технології на рівні мікроконтролерів, що використовуються в пристроях розумного будинку
- стабільність роботи технології в реальних умовах використання пристроїв
- наявність достатньої кількості периферійних датчиків і сенсорів для роботи по обраній технології
- вартість реалізації пристрою, що вплине на попит зі сторони користувачів

Провівши детальний аналіз існуючого ринку мікроконтролерів та опрацювавши існуючі аналогічні пристрої розумного дому було відхилено декілька варіантів через відносну велику вартість комплектації апаратного забезпечення, а саме X10, Isteon та KNX. Крім цього, поріг входу для налаштування даних технологій значно вище з іншими, що не дозволить не досвідченим користувачам самостійно будувати свої пристрої на основі запропонованої архітектури.

Такі технології як ZigBee та Z-Wave були відхилені через не стандартизовані протоколи обміну та передачі даних. Цей факт може ускладнити реалізацію програмного забезпечення під різні периферійні пристрої. Крім цього дані технології не дозволяють реалізувати ізольовану архітектури і потребують взаємодії багатьох пристроїв (роутерів, кінцевих точок та координаторів).

Ethernet технологія має зрозумілий інтерфейс роботи та дуже поширена в сучасних системах розумних будинків, але була відсіяна через низький користувальницький інтерфейс, пов’язано це з необхідністю провідникового зв'язку з кожним пристроєм, що уже не відповідає сучасним стандартам та трендам систем розумного будинку.

Технологія Bluetooth набуває значного поширення в сьогоднішніх процесах передачі даних і поширюється все більше в різних системах, де потрібно обмінюватись даними безпроводового зв’язку. Але на відміну технології Wi-Fi на сьогоднішній день наринку недостатньо систем які здатні спілкуватись з пристроями через Bluetooth і для цього являється необхідним наявність хабу для зв’язку з периферією, що значно ускладнює реалізацію повноцінної системи і підвищує її собівартість. Ще одним фактором відмови від Bluetooth являється недостатня доступність необхідних комплектацій наринку і їх відносна велика вартість, на відміну від Wi-Fi.

Таким чином, опрацювавши знайдені технології вибір було покладено на технологію Wi-Fi як основну для обміну даними між пристроями та системи автоматизації розумним домом. Технологія оптимальна і відповідає більшості критерії поставлених до неї. Наринку доступна необхідна кількість комплектаційних матеріалів для сбору апаратного забезпечення, а роутер для зв’язку по Wi-Fi є в кожному будинку. Крім цього Wi-Fi дозволить зручно розміщувати свої пристрої вбудь-якій точці без потреби бездротового з’єднання, і за необхідності пристрій можна перевести на альтернативні джерела енергії, яких буде достатньо для повноцінної роботи.

Висновки

В першому розділі було проведено аналіз існуючого стеку технологій, що використовуються в сучасних системах та архітектура розумного будинку. На основі виконаного аналізу було створено перелік найпопулярніших та найбільш розповсюджених протоколів передачі даних та технік, на основі яких будуються системи управління та менеджменту розумних будинків.

Згідно результатів, можно зробити висновок, що через велику кількість технології, є активна тенденція розвитку технологій розумного будинку, а отже, запити користувачів зростають разом із їх розповсюдженням. На сьогоднішній день продовжують створюватись нові технології та розвиватись існуючі, а перед користувачем постає задача вибору найбільш оптимального рішення, яке може закрити всі його задачі та потреба в домашній чи промисловій автоматизації.

На основі аналізу в роботі буде взято за основу безпроводова технологія Wi-Fi, через її простоту і розповсюдженість, а також доступності в контроллерах пристроїв розумного дому, так як більшість мають вбудовані Wi-Fi компоненти.

РОЗДІЛ 2
ОГЛЯД СУЧАСНИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ

В зв’язку з тим, що на сьогоднішній день ринок розумних пристроїв дуже великий, звичайним користувачам найскладнішою задачею являється об’єднання всіх своїх девайсів в єдину систему управління для забезпечення контроотролю на своїми об’єктами з одного місця.

Для вирішення даної задачі майже всі виробники розумного обладнання пропонують власні систему управління розумним, але тільки для девайсів власного виробництва. Такий підхід не дає можливості користувачам в повній мірі вирішувати свої задачі по автоматизації і змушує встановлювати безліч додатків для менеджемнту власного дому.

На сьогоднішній день існує величезна кількість програмного забезпечення для управління вендорськими пристроями (рис. 2.1).

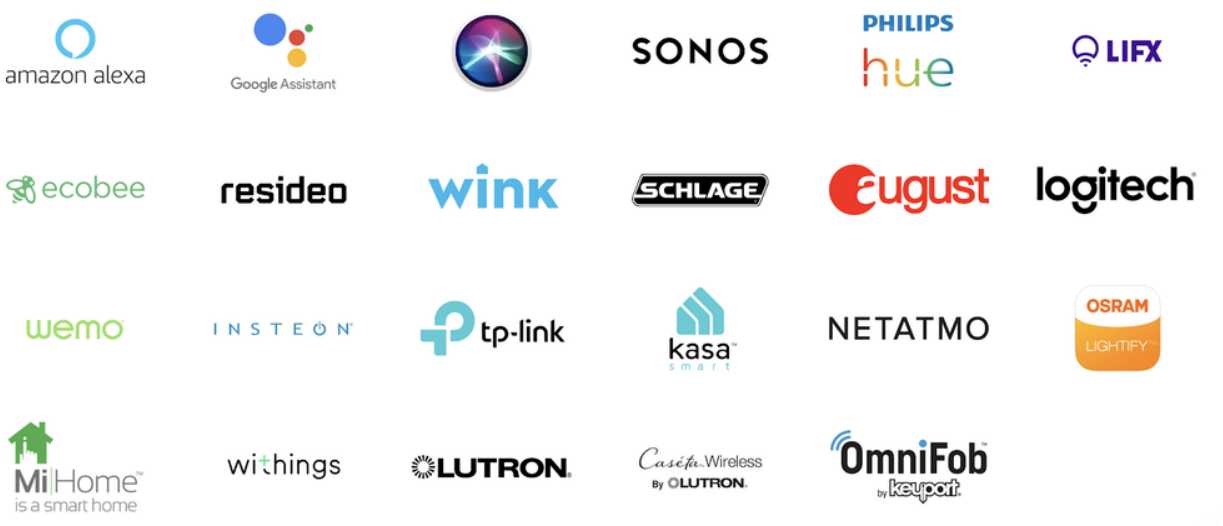


Рис. 2.1 Популярні додатки для управління пристроями розумного будинку.

2.1 Види систем управління розумним домом

Беручи до уваги, факт наявності такої величезної кількості систем управління розумним домом, важливим завдання роботи став аналіз технології, які є найбільш популярними. Для того щоб розроблена архітектура

відповідала сучасним потреба користувачів, пристрої мають бути максимально гнучкими в питання інтеграції з більшістю сучасних платформ автоматизації та відповідати сучасним потребам та завдання користувача.

Всі системи управління можна умовно розділити на дві глобальні категорії:

- Вендроське програмне забезпечення з закритою кодовою базою, протоколами передачі та власними алгоритмами роботи та фіксованою чи стандартною платою за інтеграцію девайсу.
- Open source проекти, що набули великої популярності в середовищі домашніх автоматизаторів і мають відкриту кодову базу, здатні до масштабування та нововведення нових протоколів та технології, а також являються абсолютно безкоштовними.

В даному розділі буде наведено огляд найпопулярніших шістнадцяти Open source проектів для управління розумним домом, вивчено їх основні технології та можливості та виділено ряд переваг та недоліків по їх роботі.

Переваги програмного забезпечення з відкритим вихідним кодом величезні, і багато платформ домашньої автоматизації пропонують 100% безкоштовне та функціональне програмне забезпечення ентузіастам Інтернету речей по всьому світу.

Як і більшість інших програмних платформ, гарне рішення вимагає настільки ж сильного спільноти, яка готова підтримати його і поліпшити його початковий стан.

2.2 Системи з відкритим вихідним кодом

OpenHab

OpenHAB (скорочення від Open Home Automation Bus) - один з найвідоміших інструментів домашньої автоматизації серед ентузіастів з відкритим вихідним кодом, з великим співтовариством користувачів і великою кількістю підтримуваних пристроїв і інтеграцій. Написаний на Java, openHAB переноситься в більшість основних операційних систем і навіть добре працює

на Raspberry Pi. Підтримуючи сотні пристроїв, openHAB не пристроїв і спрощує розробникам додавання власних пристроїв або модулів в систему. OpenHAB також поставляє додатки iOS і Android для управління пристроями, а також інструменти дизайну, щоб ви могли створити свій власний користувальницький інтерфейс для своєї домашньої системи (рис. 2.2).

Рис. 2.2 Платформа OpenHab для домашньої автоматизації

OpenHAB - це система домашньої автоматизації з відкритим вихідним кодом, яка також має потужне підтримує спільнота. OpenHAB має архітектуру з підтримкою плагінів, яка допомагає розробникам додавати нові пристрої або інтегрувати нові сервіси. Він також має зручний для розробників REST-API, індивідуальний дизайн з тригерами на основі часу і подій, службу повідомлень і управління на основі VoiceUI.

OpenHAB працює в Linux, Windows і macOS, але також може працювати в Raspberry PI, Pine64 і Docker. Він пропонує мобільні додатки для пристроїв на базі Android і додаток iOS для iPhone і iPad.

З OpenHAB користувачеві не потрібно запускати хмара, якщо користувачеві потрібно обмежена система для захисту його конфіденційності, але розробники гарантують, що це дружня до хмари система, якщо цього вимагає користувач. Коротше кажучи, OpenHAB може підтримувати Google Assistant, Amazon Alexa, IFTTT і Apple HomeKit. Щоб OpenHAB був готовий до роботи в хмарі, OpenHAB пропонує хмарну версію, яка може бути розміщена на їх сервісі, або користувач може розмістити її на своєму сервері. Він має понад 1500 підтримуваних речей [10].

OpenHAB випускається під Eclipse Public License і написаний на Java. Ви можете завантажити це тут. Вихідний код OpenHAB можна скачати з GitHub.

Переваги OpenHab:

1. Велика спілка користувачів
2. Понад 200 інтеграції з різними пристроями
3. Відкрите API для зовнішніх розробників та DIY користувачів
4. Власний WEB інтерфейс та мобільні додатки
5. Відкрита інтеграція через протокол MQTT

Calaos

Calaos розроблений як повнофункціональна платформа домашньої автоматизації, що включає серверний додаток, інтерфейс з сенсорним екраном, веб-додаток, власні мобільні додатки для iOS і Android, а також попередньо налаштовану операційну систему Linux для роботи під нею. Проект Calaos був створений французькою компанією, тому її форуми підтримки в основному ведуться на французькій мові, хоча велика частина інструкцій і документації переведена на англійську.

Він розроблений як повнофункціональна платформа домашньої автоматизації з відкритим вихідним кодом, включаючи серверний додаток, інтерфейс з сенсорним екраном, веб-додаток, власні мобільні додатки для iOS і Android, а також попередньо налаштовану операційну систему Linux для

роботи під нею. Проект Calaos був створений французькою компанією, тому її форуми підтримки в основному ведуться на французькій мові, хоча велика частина інструкцій і документації переведена на англійську. Calaos підтримує безліч: Wago PLC, Raspberry Pi, Zodianet's ZiBASE, Cubieboard, Squeezebox, CCTV і багато інших, постійно додаючи все більше і більше апаратних платформ (рис. 2.3).

Рис. 2.3 Платформа Calaos - повнофункціональна платформа для домашньої автоматизації

Domoticz

Domoticz - це система домашньої автоматизації з відкритим вихідним кодом з повністю широкої бібліотекою підтримуваних пристроїв, починаючи від метеостанцій і закінчуючи детекторами диму і пультами дистанційного керування, і велика кількість додаткових сторонніх інтеграцій задокументовано на веб-сайті проекту.

Він планується з інтерфейсом HTML5, що робить його доступним з програм робочої області і більшості сучасних мобільних телефонів, і є легким, працює на деяких пристроях з низьким рівнем управління, таких як Raspberry Pi. Domoticz написаний в основному на C / C ++ під ліцензією GPLv3.

Domoticz - це система домашньої автоматизації з досить широкою бібліотекою підтримуваних пристроїв, починаючи від метеостанцій і закінчуючи детекторами диму і пультами дистанційного керування, і велика кількість додаткових сторонніх інтеграцій задокументовано на веб-сайті проекту. Він розроблений з використанням інтерфейсу HTML5, що робить його доступним для настільних браузерів і більшості сучасних смартфонів, і є легким, працює на багатьох малопотужних пристроях, таких як Raspberry Pi (рис. 2.4).

Рис. 2.4 Платформа Domoticz для домашньої автоматизації з відкритим вихідним кодом

Home Assistant

Home Assistant - це платформа домашньої автоматизації з відкритим вихідним кодом, призначена для простого розгортання практично на будь-якій машині, на якій може працювати Python 3, від Raspberry Pi до пристрою мережевого зберігання (NAS), і вона навіть поставляється з контейнером Docker для розгортання. на інших системах - легкий вітерець. Він інтегрується з великою кількістю пропозицій з відкритим вихідним кодом, а також з комерційними пропозиціями, дозволяючи зв'язати, наприклад, IFTTT, інформацію про погоду чи ваш пристрій Amazon Echo для управління обладнанням від замків до освітлення. Home Assistant випущений під ліцензією MIT.

Home Assistant - це платформа домашньої автоматизації з відкритим вихідним кодом, призначена для простого розгортання практично на будь-якій машині, на якій може працювати Python 3, від Raspberry Pi до пристрою мережевого зберігання (NAS), і вона навіть поставляється з контейнером Docker для розгортання. на інших системах - легкий вітерець. Він інтегрується з великою кількістю пропозицій з відкритим вихідним кодом, а також з комерційними пропозиціями, дозволяючи зв'язати, наприклад, IFTTT, інформацію про погоду чи ваш пристрій Amazon Echo для управління обладнанням від замків до освітлення [11].

Home Assistant забезпечує велику і безшовну інтеграцію з Amazon Alexa, Google Assistant і голосовим помічником Mycroft.io з відкритим вихідним кодом(рис. 2.5).

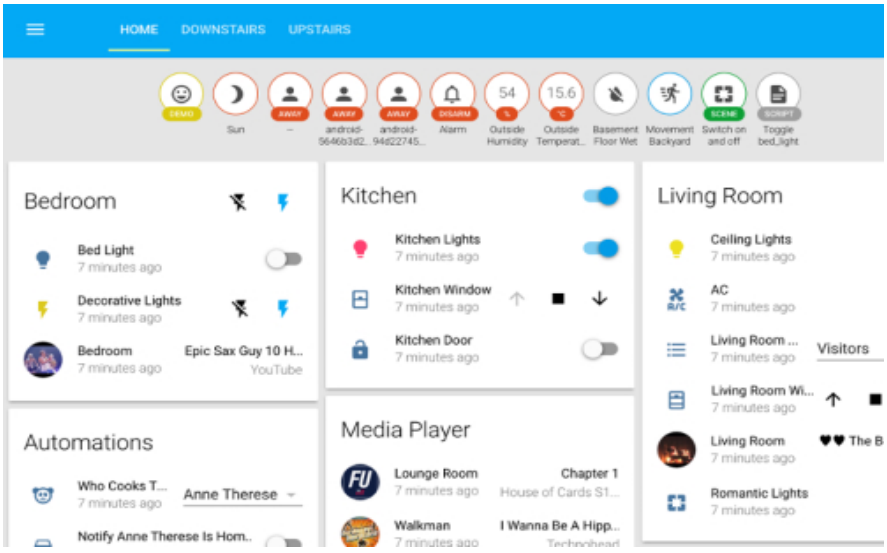


Рис. 2.5 Платформа Home Assistant для домашньої автоматизації з відкритим вихідним кодом

Mister House

MisterHouse значно поширився з 2016 року, він використовує сценарії Perl для відстеження всього, що може бути запрошено комп'ютером, або управління чим-небудь, яким можна керувати віддалено. Він реагує на голосові команди, час дня, погоду, місце розташування та інші події, щоб включити світло, розбудити вас, записати ваше улюблене телешоу, оголосити,

що дзвонять по телефону, попередити, що ваші вхідні двері відкриті, повідомити, скільки часу ваш син був в мережі, повідомити вам, перевищує швидкість машина вашої дочки, і багато іншого. Він працює на комп'ютерах Linux, macOS і Windows і може читати / писати з самих різних пристроїв, включаючи системи безпеки, метеостанції, ідентифікатори абонента, маршрутизатори, системи визначення місцезнаходження транспортних засобів та багато іншого [12].

Він працює на комп'ютерах Linux, macOS і Windows і може читати / писати з самих різних пристроїв, включаючи системи безпеки, метеостанції, ідентифікатори абонента, маршрутизатори, системи визначення місцезнаходження транспортних засобів та багато іншого (рис. 2.6.)

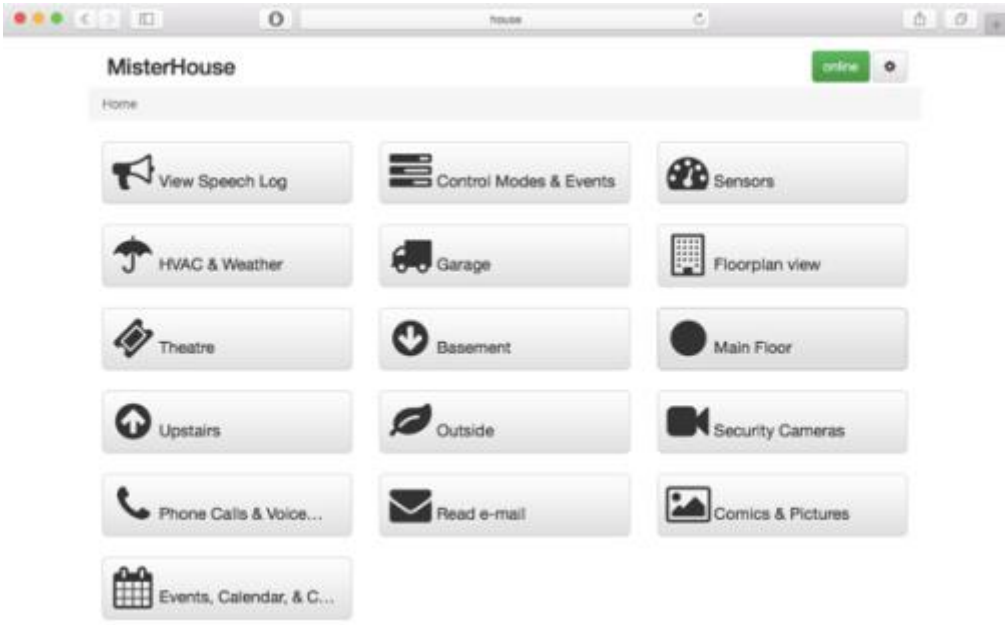


Рис. 2.6 Платформа Mister House для домашньої з використанням сценарію Perl

Open Motics

OpenMotics - це система домашньої автоматизації з апаратним і програмним забезпеченням під ліцензіями з відкритим вихідним кодом. Він розроблений для забезпечення комплексної системи управління пристроями, а не для об'єднання безлічі пристроїв від різних постачальників. На відміну від

багатьох інших систем, розроблених в першу чергу для легкої модернізації, OpenMotics фокусується на дротовому вирішенні. Для отримання додаткової інформації див. Нашу повну статтю від Backend розробника OpenMotics Фредеріка Рікбоша.

OpenMotics призначений для забезпечення комплексної системи управління пристроями, а не для об'єднання безлічі пристроїв від різних постачальників. Платформа OpenMotics поєднує в собі доступне обладнання з відкритим вихідним кодом і сучасні хмарні рішення [13].

Легко налаштовуйте, відстежуйте і використовуйте кожен аспект вашої установки за допомогою наших інтуїтивно зрозумілих інтерфейсів на ПК, планшеті і смартфоні. На відміну від багатьох інших систем, розроблених в першу чергу для легкої модернізації, OpenMotics фокусується на дротовому вирішенні (рис. 2.7.).

Рис. 2.7 Платформа Open Motics з апаратним і програмним забезпеченням під ліцензіями з відкритим вихідним кодом

Home Genie

HomeGenie - це сервер з відкритим вихідним кодом для домашньої автоматизації, він створений для управління, контролю, моніторингу та автоматизації пристроїв, підключених до Інтернету (інтелектуальних

пристроїв). Він забезпечує безшовну інтеграцію з багатьма пристроями і службами, а також підтримує безліч протоколів і технологій, таких як IR / RF Control і UPnP / DLNA.

Розроблений на Мультістандартний основі, HomeGenie може взаємодіяти з різними пристроями, такими як X10, Insteon, Z-Wave, Philips Hue, UPnP / DLNA, RFXCom, KNX, взаємодіяти з зовнішніми веб-службами та інтегрувати все це в загальну середу автоматизації. Таким чином, навіть якщо вони засновані на різних стандартах, всередині HomeGenie все «модулі» можна контролювати і автоматизувати, щоб вони працювали разом. Завдяки сучасному вбудованому веб-інтерфейсу користувача, HomeGenie можна використовувати з будь-якого ПК, смартфона або планшета [14].

HomeGenie можна встановити в Windows, Linux і macOS. Підтримує Raspberry Pi. Його досить легко встановити, налаштувати і керувати навіть для початківців користувачів (рис. 2.8.).

Рис. 2.8 Платформа Home Genie з відкритим вихідним кодом для домашньої автоматизації

JeeDom

Jeedom - це безкоштовне програмне забезпечення з відкритим вихідним кодом, яке можна встановити в будь-якій системі Linux, включаючи Raspberry

Pi. Він заснований на ядрі і має безліч функцій: просте і розширене управління сценаріями, взаємодія тексту і звуку з системою домашньої автоматизації, перегляд історії, створення кривих і графіків, зв'язування всього обладнання і підключених об'єктів, персоналізація інтерфейсу.

Jeedom, інструмент домашньої автоматизації з відкритим вихідним кодом, може управляти багатьма інтелектуальними пристроями, такими як ліхтарі, макети і навіть мультимедійні пристрої. Для управління на ходу є мобільні додатки для Android і iOS. Примітно, що Jeedom дійсно пропонує готове рішення для пристрою, яка готова до використання прямо з коробки.

З Jeedom ви можете:

- Керувати безпекою товарів і людей
- Автоматизувати опалення для більшого комфорту і економії енергії
- Перегляд і управління використанням енергії для прогнозування витрат і скорочення використання
- Спілкуйтеся за допомогою голосу, SMS, електронної пошти або мобільних додатків
- Керуйте всіма автоматичними пристроями будинку: віконницями, воротами, освітленням і т. Д.
- Управління мультимедійними аудіо і відео пристроїв, а також підключеними об'єктами.

Jeedom - відмінний варіант для домашньої автоматизації, яка, на жаль, стримується мовою. Спільнота Jeedom переважно французьке, і навіть якщо веб-сайт перекладено англійською, правила спільноти і форуми все на французькому. Що стосується документації, Jeedom пропонує безліч мов, включаючи англійську, іспанську та німецьку (рис. 2.9.). Однак це може бути трохи складніше в порівнянні з платформами домашньої автоматизації, такими як openHAB і Home Assistant [15].

Рис. 2.9 Платформа JeeDom програмне забезпечення з відкритим вихідним для систем Linux

FreeDomotic

Freedomotic - це гнучкий і безпечний фреймворк для розробки Інтернету речей (IoT) з відкритим вихідним кодом. Його можна використовувати для створення сучасних розумних просторів і управління ними. Він націлений на приватних осіб (домашня автоматизація), а також на підприємства (інтелектуальна роздрібна среда, маркетинг з урахуванням зовнішнього середовища, моніторинг і аналітика і т. Д.). Freedomotic може взаємодіяти з відомими протоколами автоматизації, а також з рішеннями для самостійної роботи. Він розглядає Інтернет, соціальні мережі і фірмові інтерфейси як першокласні компоненти системи.

Freedomotic може бути інтегрований з популярними технологіями автоматизації будівель, такими як BTicino OpenWebNet, Modbus RTU, Z-wave, а також до призначених для користувача проектами автоматизації з використанням пристроїв Arduino, саморобних плат, сторонніх графічних інтерфейсів, механізмів перетворення тексту в мову, виявлення руху з використанням потоку IP-камер, соціальні мережі та багато іншого ... Всі ці

функції можуть бути доставлені з торговельного майданчика у вигляді завантажуваних плагінів.

Freedomotic написаний на Java. Таким чином, він може працювати в Windows, Linux, Mac і Solaris. Підтримує Raspberry Pi. Freedomotic доступний більш ніж на 20 мовах (рис. 2.10).

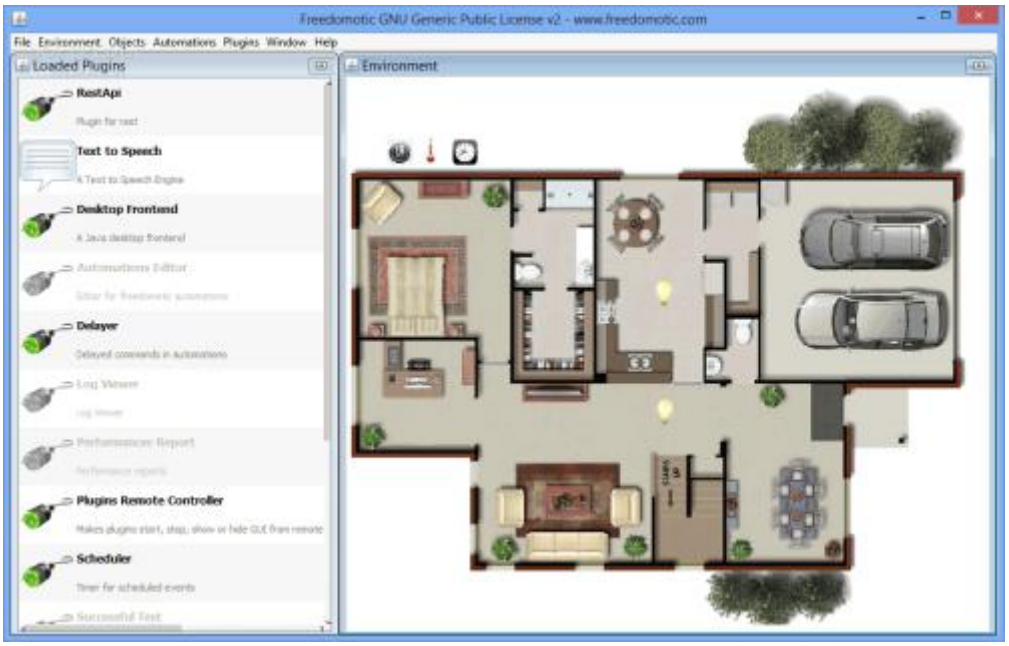


Рис. 2.10 Платформа Freedomotic - гнучкий і безпечний фреймворк

ioBroker

ioBroker - це платформа автоматизації з відкритим вихідним кодом, написана за допомогою NodeJS і працює на Windows, Linux, macOS і SBC «Одноплатні комп'ютери», такі як Raspberry Pi. Це повноцінна платформа для IoT «Інтернету речей», яка з легкістю підтримує додавання, налаштування і управління пристроями.

ioBroker має сотні адаптерів, які нагадують сервіси, пристрої, інші платформи, датчики, системи безпеки і протоколи [16].

ioBroker - це платформа Інтернету речей на основі JavaScript, яка може керувати освітленням, замками, термостатами, мультимедіа, веб-камерами і т. д. Він знаходиться під ліцензією MIT (рис 2.11).

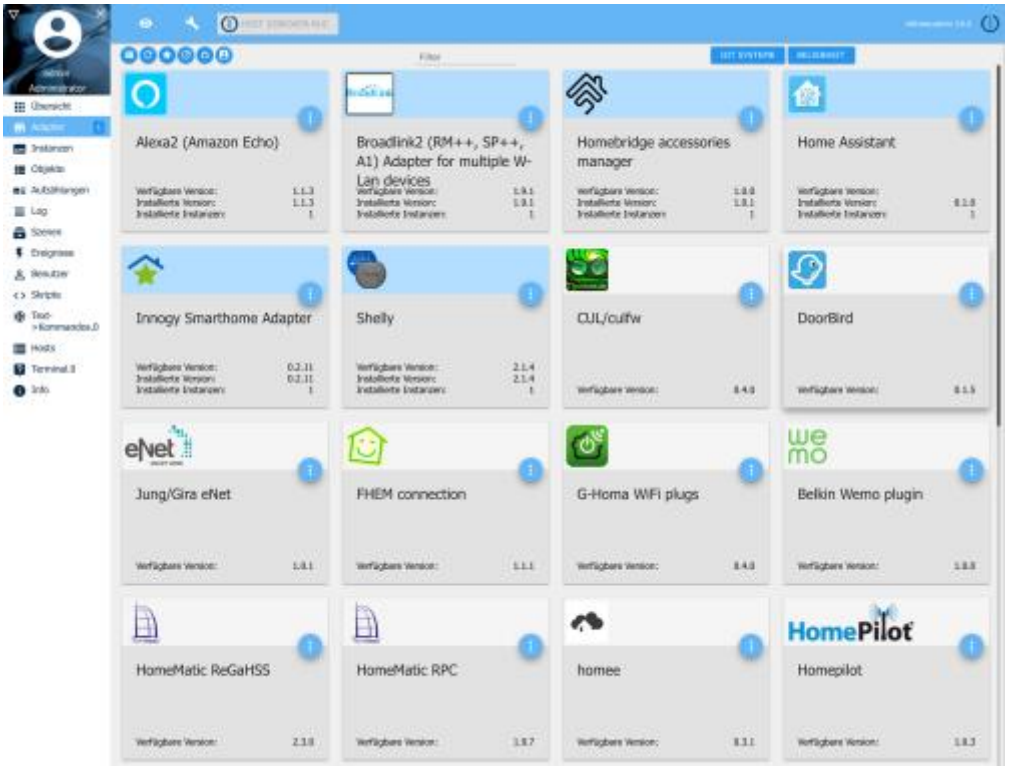


Рис. 2.11 Платформа ioBroker відкритим вихідним кодом для систем Windows, Linux, macOS і SBC

fHem

FHEM - це Perl-сервер під GPL для домашньої автоматизації. Він використовується для автоматизації деяких загальних завдань в будинку, таких як перемикання ламп / жалюзі / опалення і т. Д., А також для реєстрації подій, таких як температура / вологість / споживання енергії. Програма працює як сервер, ви можете керувати ним через веб-інтерфейс або інтерфейс смартфона, telnet або TCP / IP безпосередньо.

Щоб використовувати FHEM, вам знадобиться цілодобовий сервер (NAS, RPi, ПК, MacMini і т. Д.) (рис. 2.12).

Основні особливості:

- Підтримка безлічі протоколів, використовуваних в домашньої автоматизації, аудіо / відео пристроях, погодних службах, онлайн-календарях і т. Д.

- автоматичне створення пристроїв / журналів при отриманні даних з нового пристрою: запустіть FHEM і подивіться, як автоматично з'являються графіки ваших датчиків.
- запис подій в файли або базу даних за допомогою фільтрів регулярних виразів
- повідомлення зовнішніх програм або скриптів при отриманні певних подій
- команди по часу (наприклад, включення лампи з заходу до півночі)
- безліч інтерфейсів: простий текст, JSON, XML, кожен з них за звичайним TCP / IP, SSL або HTTP.
- модульна архітектура з нині понад 430 модулів, легко додати ваше спеціальний пристрій багато зовнішніх інтерфейсів [17]

Рис. 2.12 Платформа fHem домашньої автоматизації

OpenNetHome

OpenNetHome - це програмно-апаратна система з відкритим вихідним кодом для домашньої автоматизації. Він заснований на Java і Apache Maven і працює в системах Windows, macOS і Linux. Його також можна встановити на

Raspberry Pi. Він пропонує веб-панель для легкого контролю, управління, моніторингу та автоматизації функцій розумного будинку.

За допомогою цієї платформи розумного будинку з відкритим вихідним кодом ви можете контролювати і управляти багатьма пристроями, включаючи диммери, пожежну сигналізацію, термометри, датчики вологості і багато іншого. OpenNetHome випущений під GPLv3 (рис. 2.14) [18].

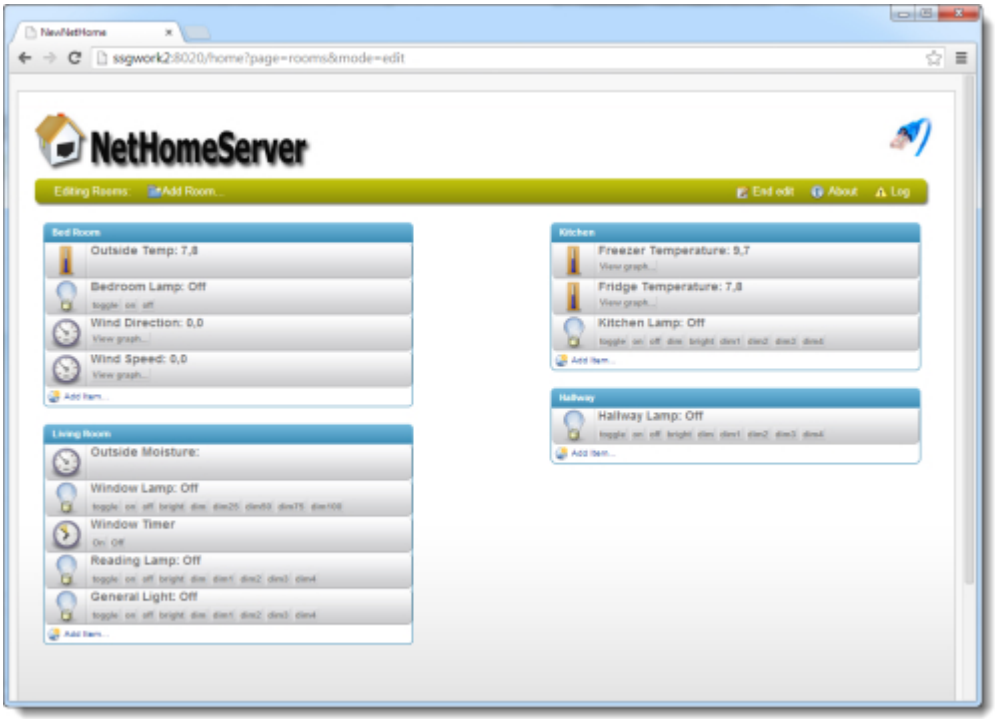


Рис. 2.14 Платформа OpenNetHome як програмно-апаратна система з відкритим вихідним кодом для домашньої автоматизації

AGO Control

AGO Control - це система домашньої автоматизації за ліцензією GPLv3, створена для Raspberry Pi і вбудованих систем. Він також пропонує міст-зв'язок з іншими системами розумного будинку, такими як LinuxMCE.

AGO Control підтримує безліч інтелектуальних і мультимедійних пристроїв, таких як Z-Wave, 1wire, KNX / EIB, MySensors, Chromoflex USP3, AV-ресивери Onkyo і багато інших (рис. 2.15).

Основні особливості:

- використовує шину повідомлень AMQP Enterprise в якості комунікаційного сервера
- легкий протокол, якого читають людьми і машинами
- сучасна і модульна архітектура
- він поставляється з хмарними функціями
- схема пристрою визначена в YAML
- відмінна продуктивність - також працює на вбудованих пристроях, таких як Raspberry Pi, і на кількох підключаються комп'ютерах, таких як Sheevaplug і Guruplug
- легко розширити
- підтримка багатьох пристроїв і протоколів

Підтримує Raspberry Pi, а також працює з Android [19].

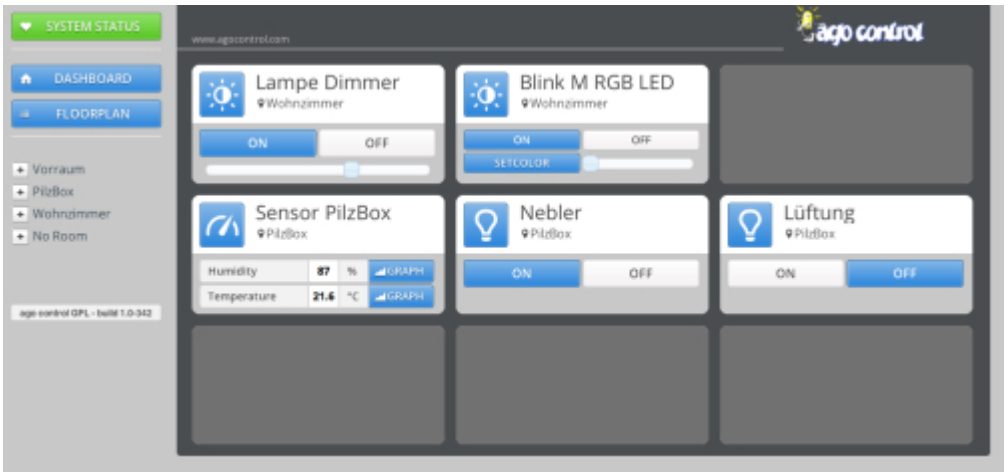


Рис. 2.15 Платформа AGO Control для домашньої автоматизації в вбудованих систем

HomeBridge

Homebridge дозволяє інтегруватися з пристроями розумного будинку, які спочатку не підтримують HomeKit. Існує більше 2000 плагінів Homebridge, що підтримують тисячі різних інтелектуальних аксесуарів.

Платформа в більшій мірі призначена для інтеграції пристроїв розумного дому з іншими платформами, а саме HomeKit — екосистема компанії Apple, яка дає можливість керувати пристроями за допомогою голосового асистента Siri. Крім цього платформа має в своєму арсеналі безліч додаткових плагінів для всіх популярних асистентів, а саме Google, Alexa, Alisa та інші.

piDome

piDome - це платформа домашньої автоматизації на базі RaspberryPi. Його мета - надати просту у використанні систему для нетехнічних фахівців. piDone підтримує мультимедійну систему MySensors, RFXCOM, KODI. Він пропонує сервер, на якому, як ми вже згадували, працює RaspberryPi, і клієнти для різних систем.

За допомогою piDome користувач може створювати і налаштовувати свою панель керування, використовувати монітори з різним дозволом і запускати клієнти piDome в Windows, Android і macOS. Він пропонує рішення як для кінцевих користувачів, так і для розробників і любителів (рис. 2.16) .

Крім того, що він є сервером, він також включає клієнтів для декількох платформ.

Основні особливості:

- Підходить для технічних і нетехнічних користувачів
- Одночасне виконання декількох команд
- Дашборда для всіх типів клієнтів
- Брокер MQTT з клієнтськими функціями
- Модулі для: комунальних вимірювань, універсального пульта дистанційного керування, SMS, мультимедіа (XBMC) і даних про погоду
- Автоматичне створення графіків даних
- Дозволяє написати свій власний клієнт з клієнтськими бібліотеками Java [20]

Рис. 2.16 Платформа piDome для домашньої автоматизації на базі RaspberryPi

Pimatic

Pimatic - це фреймворк для домашньої автоматизації, що працює на node.js. Він надає загальну платформу для задач управління будинком та автоматизації. Плагін мобільного інтерфейсу надає приємний веб-інтерфейс з оглядом датчиків, управлінням пристроями та визначенням правил. Веб-інтерфейс побудований з використанням Express і jQuery Mobile.

Pimatic поставляється з архітектурою, готової до плагінів, яка дозволяє розробникам додавати пристрої, служби і протоколи (рис. 2.17).

Основна увага в цій структурі приділяється гнучкості: її можна використовувати досить швидко і «легко». Завдяки вбудованим функціям ви можете відразу автоматизувати завдання, підключивши домашні пристрої і додавши умовні правила. Підтримує Raspberry Pi.

Рис. 2.17 Платформа Pimatic як фреймворк для домашньої автоматизації

MyController

MyController - це платформа для домашньої автоматизації та Інтернету речей з відкритим вихідним кодом. Спочатку він почав підтримувати проект MySensors. Він розроблений для роботи на машині з обмеженими вимогами до устаткування. Оскільки він побудований на Java, він працює на «одноплатних комп'ютерах» Windows, Linux, macOS і SBC, включаючи Raspberry Pi, Pine64, Rock64, Orange Pi.

MyController підтримує безліч мереж, шлюзів, протоколів і пристроїв. Це сама сумісна система домашньої автоматизації з MySensors в цьому списку (рис. 2.18).

Підтримує декілька шлюзів з декількома протоколами: Serial, Ethernet і MQTT.

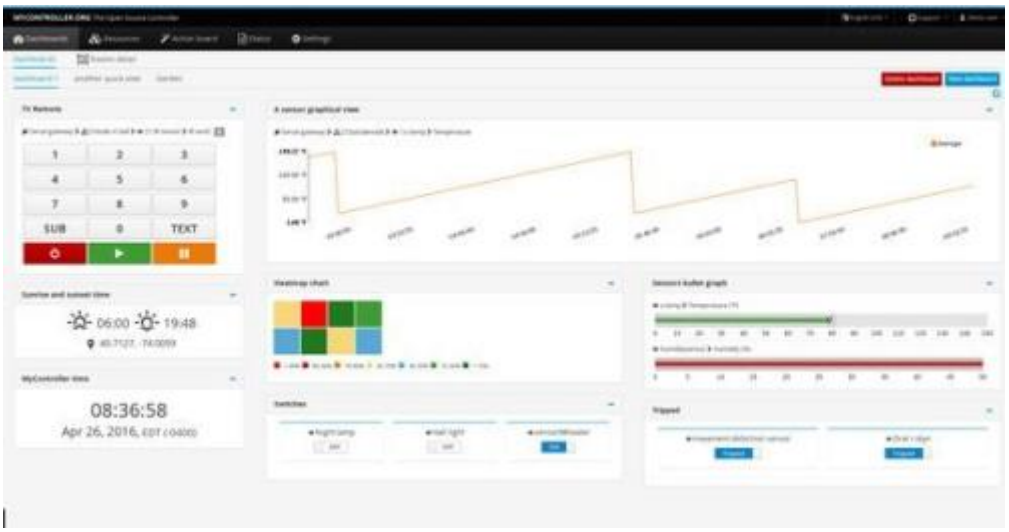


Рис. 2.18 Платформа MyController для домашньої автоматизації та Інтернету речей з відкритим вихідним кодом

SmartHomatic

Smarthomatic (SHC) - це система домашньої автоматизації з відкритим вихідним кодом, що забезпечує основу для створення більшої мережі датчиків і виконавчих механізмів, які працюють разом. Він складається з декількох повністю розроблених апаратних пристроїв, які ви можете створити самостійно, і програмного забезпечення, що працює на цих пристроях.

Цей проект зосереджений на апаратних пристроях і їх програмному забезпеченні (прошивці), а не на призначених для користувача інтерфейсів для управління такими завданнями, як веб-інтерфейси або додатки для смартфонів. Проте, ця система призначена для легкої інтеграції в такі системи (рис. 2.19).

Ліцензований під GPLv3, він використовується для таких речей, як управління освітленням, приладами і вологістю повітря, вимірювання температури навколишнього середовища і не забувайте поливати рослини.

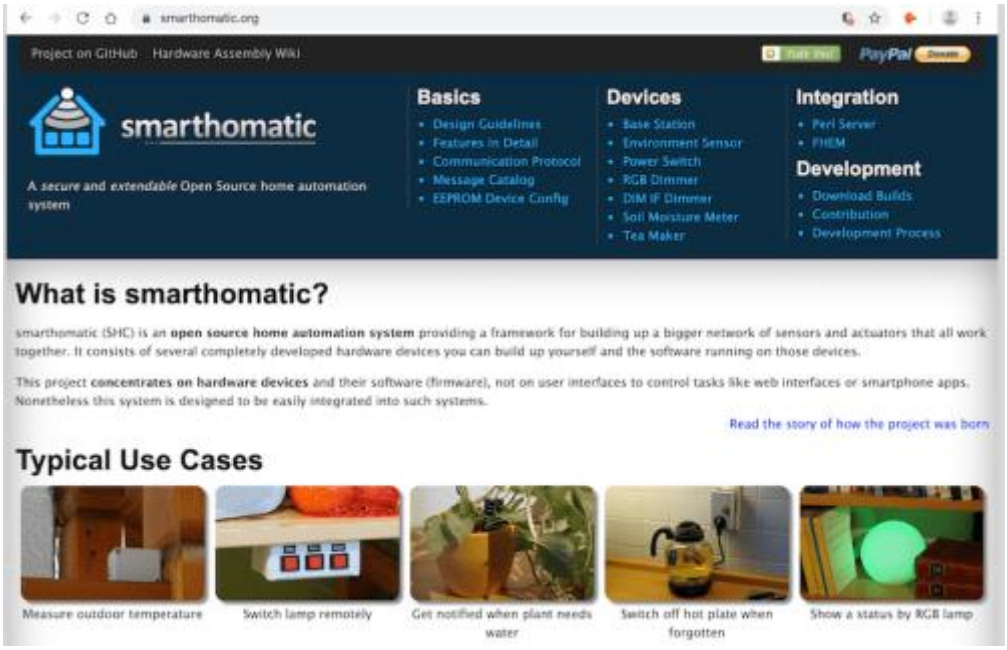


Рис. 2.19 Платформа Smarthomatic домашньої автоматизації з відкритим вихідним кодом

EventGhost

EventGhost - це популярна програма домашньої автоматизації з відкритим вихідним кодом (GPL v2) для Windows, яка використовується багатьма в співтоваристві для автоматизації простих завдань. Завдяки цій простій структурі ви можете створювати так звані набори завдань, з одного боку шляху автоматизації, через які проходять ваші пристрої.

Одна з найкращих особливостей EventGhost полягає в тому, що він розширюваний і має більше 300 доступних унікальних розширень (плагінів). Деякі приклади пристроїв, з якими може взаємодіяти EventGhost, - це Yamaha, Denon / Marantz, Pioneer, Sony, Sonos, Samsung, RTI, Amazon Echo, MicasaVerde Vera, Phillips Hue і багато інших. Підтримувані протоколи автоматизації включають, крім іншого, MQTT, IFTTT, TCP / IP, послідовний (RS-232), CIR (ІК-пульти), X10, xAP, xPL, практично будь-який розмовний людську мову. EventGhost може навіть автоматизувати завдання на комп'ютері, на якому він встановлений. Такі речі, як операції з клавіатурою, руху миші, відкриття і закриття запущених програм, зміна налаштувань звуку

і відео, а також включення і виключення моніторів. Деякі приклади подій, які запускаються через різних змін на комп'ютері, включають підключення або відключення USB-пристрою, коли він переходить з розетки на харчування від батареї (ІБП або ноутбук), вхід в систему або виключення, запуск і зупинка сеансів віддаленого робочого стола , ви може навіть EventGhost ініціювати подія, якщо файл був змінений. Подібні здатності роблять його найпотужнішим інструментом у вашому арсеналі домашньої автоматизації.

EventGhost дозволяє користувачам управляти мультимедійними ПК і підключеним обладнанням за допомогою плагінів, що запускають макроси, або шляхом написання власних скриптів Python (рис. 2.20) [21].

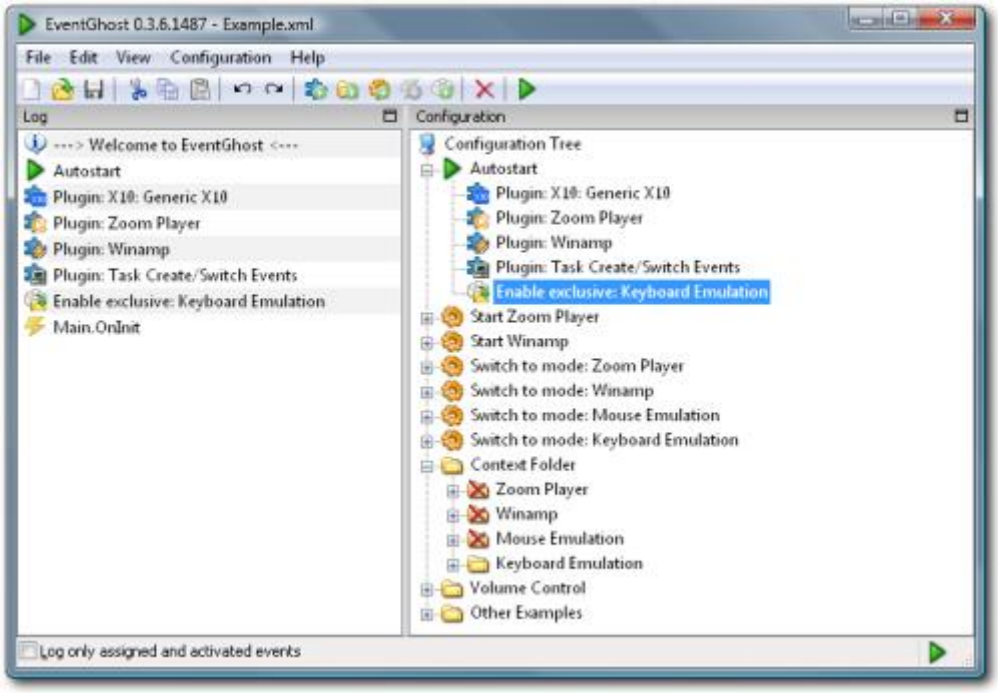


Рис. 2.20 Платформа EventGhost програма домашньої автоматизації з відкритим вихідним кодом (GPL v2) для Windows

MajorDoMo

MajorDoMo - це платформа домашньої автоматизації з відкритим вихідним кодом, призначена для використання в многопротокольной і мультисервісної середовищі. Доступно для ОС Windows і Linux. Він має

підтримку декількох брендів і декількох протоколів (MQTT, ZWave, 1-wire, ModBus, SNMP, Ethernet).

Він створений групою розробників з Росії, тому він все ще розширює свої території на міжнародний ринок з кращою підтримкою англійською мовою.

Інтерфейс являє собою веб-інтерфейс, простий у використанні для нетехнічних користувачів. Користувач може додавати кімнати, пристрої, налаштовувати автоматичні події, інтегрувати веб-додатки і хмарні сервіси (рис. 2.21) [22].

Рис. 2.21 Платформа MajorDoMo для домашньої автоматизації з відкритим вихідним кодом для використання в многопротокольному та мультисервісному середовищі

LinuxMCE

LinuxMCE або Linux Media Center Edition діє як міст між медіа і технологіями розумного будинку. LinuxMCE - це повний пакет домашньої

автоматизації, створений як дистрибутив Linux, що працює на Raspberry Pi. Він поставляється з попередньо створеним і попередньо налаштованим сервером і системою користувальницького інтерфейсу, яка забезпечує просту інтеграцію з багатьма службами пристроїв з підтримкою багатьох протоколів інтелектуальних пристроїв, таких як VoIP, QoS, HDMI та інших(рис. 2.22).

Рис. 2.22 Платформа LinuxMCE, взаємодія між медіа і технологіями розумного будинку

Gladys

Gladys - це легка домашня автоматизація, написана за допомогою NodeJS, яка працює в Windows, Linux, macOS і Raspberry Pi. Gladys3 - поточна стабільна версія, але вона несумісна з Raspberry Pi. Gladys4 Alpha в даний час знаходиться на стадії тестування.

Gladys поки що являє собою армійський проект однієї людини, але він починає привертати увагу розробників і, сподіваюся, скоро отримає більшу підтримку спільноти (рис. 2.23).

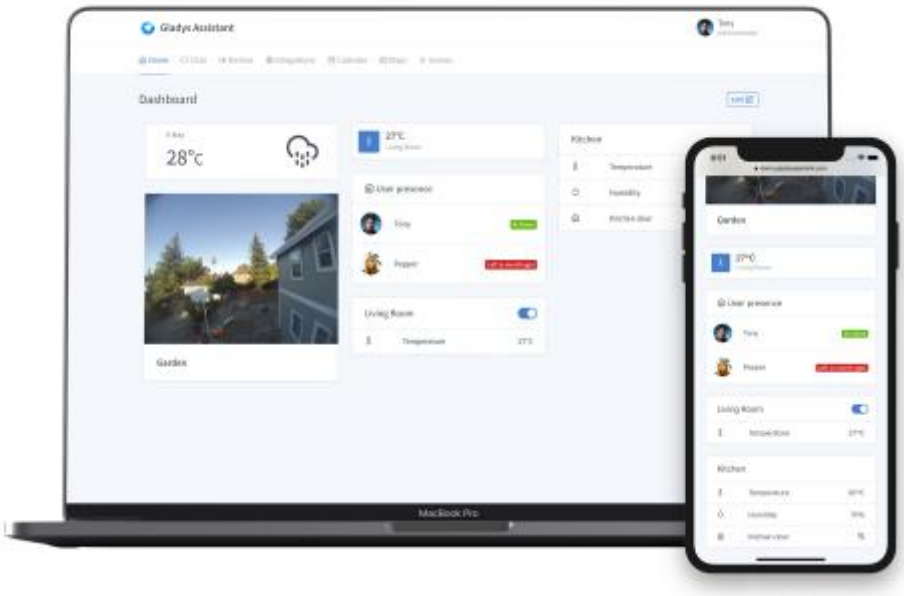


Рис. 2.23 Платформа Gladys для домашньої автоматизація, підтримка систем Windows, Linux, macOS і Raspberry Pi

The Thing System

Система Thing - це набір програмних компонентів і мережевих протоколів. Наше програмне забезпечення Steward написано на node.js, що робить його портативним і легко розширюваним. Він може працювати на вашому ноутбуці або на невеликому Одноплатний комп'ютері, такому як Raspberry Pi.

Керуючий знаходиться в центрі системи і підключається до речей у вашому домі, будь то медіаплеєри, такі як Sonos або Apple TV, термостат Nest, система управління будинком INSTEON або лампочки Philips Hue - будь то ваші речі з'єднуються разом через Wi-Fi, USB або Bluetooth Low Energy (BLE) (рис. 2.24).

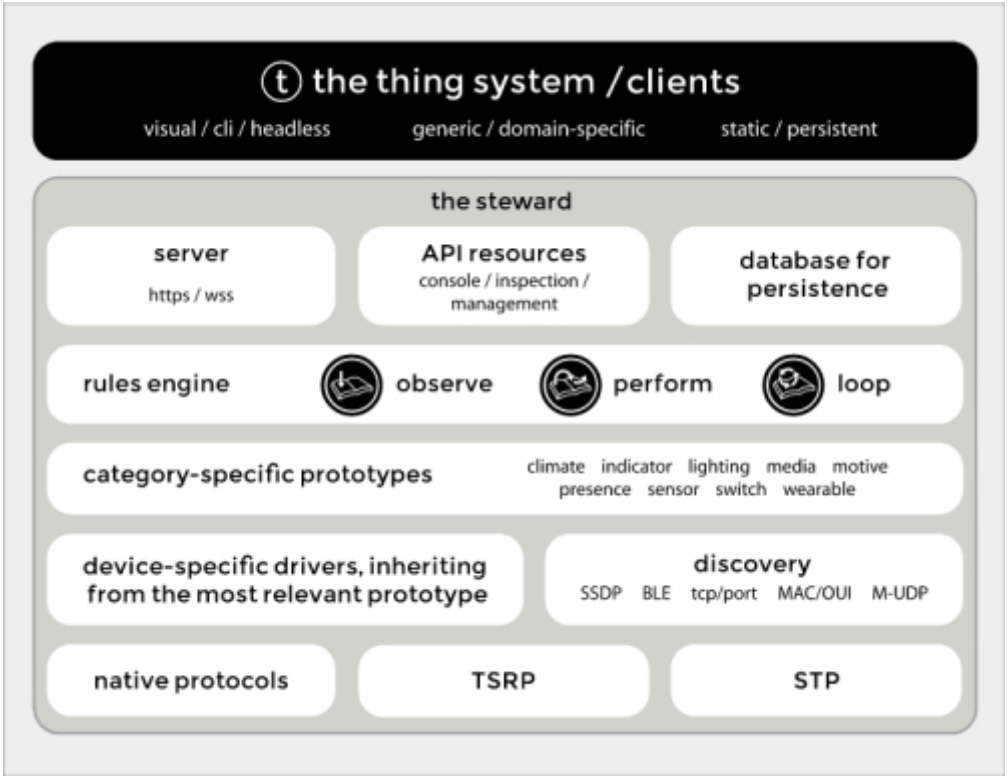


Рис. 2.24 Платформа Thing

Pytomation

Pytomation - це легка система автоматизації з відкритим вихідним кодом, написана на Python. Це програмне забезпечення можна використовувати для домашньої автоматизації та управління освітленням, хоча воно, звичайно ж, не обмежується тільки цими застосуваннями. Він може працювати на будь-якій платформі, що підтримує Python (Windows, Mac OS X, Linux і т. Д.) (рис. 2.25).

Основні особливості:

- виконує дії на основі голосового введення, часу доби, даних файлу, даних послідовного порту і даних сокета, а також послідовних і мовних протоколів.
- REST API
- клієнти для мобільного Інтернету і Android з постійним оновленням пристроїв (веб-сокети)
- голосові команди з Android (додаток Home Control)

- локальний телнет і веб-доступ
- володіє унікальною мовою для опису пристроїв і дій
- працює зі смарт-об'єктами, дверима, освітленням, датчиками руху, фотоелементами і т. д.
- додаткове програмування «міні-петлі» для більш складних елементів управління
- легке додавання нових драйверів обладнання
- добре спілкування з прикладами
- зіставляє одну команду з іншого з джерелом і часом
- додаткове програмування, кероване подіями, для складних дій при зміні стану пристрою.

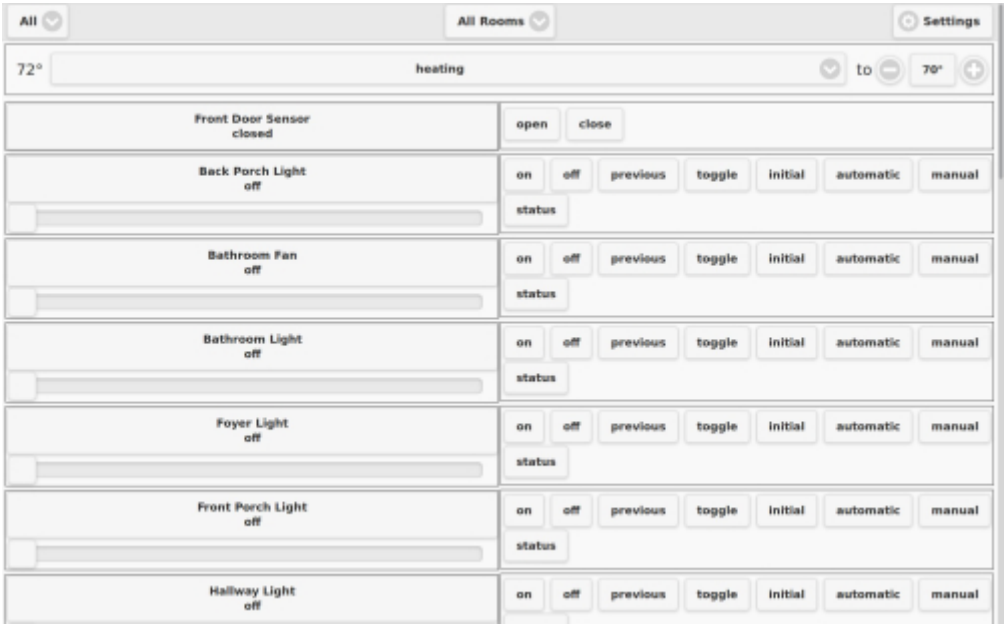


Рис. 2.25 Платформа Pytomation − система автоматизації з відкритим вихідним кодом, написана на Python

Eclipse Smart Home

Eclipse SmartHome - це платформа для створення рішень для розумного будинку. Володіючи дуже гнучкою архітектурою, він підтримує модульність, що надається OSGi для додатків Java. Таким чином, Eclipse SmartHome складається з багатого набору пакетів OSGi, які служать для різних цілей. Не

всі рішення, засновані на Eclipse SmartHome, зажадають всіх цих пакетів - замість цього вони можуть вибрати, які частини їм цікаві (рис. 2.26).

Існують наступні категорії комплектів:

- config: все, що пов'язано із загальною конфігурацією системи, наприклад файли конфігурації, синтаксичний аналіз XML, виявлення і т. д.

- core: основні зв'язки для логічної роботи системи - на основі абстрактних понять елемента і подій.

- io: всі види додаткових функцій, пов'язаних з введенням-висновком, такі як консольні команди, підтримка звуку або зв'язок HTTP / REST.

модель: підтримка предметно-орієнтованих мов (DSL).

дизайнер: Eclipse RCP підтримує DSL і інші файли конфігурації.

- ui: пов'язані з призначеним для користувача інтерфейсом пакети, які надають послуги, які можуть використовуватися різними користувача інтерфейсами, наприклад діаграми або значки [23].

Рис. 2.26 Платформа Eclipse SmartHome для створення рішень для розумного будинку

NODE-RED

Node-RED - це інструмент програмування IoT для об'єднання апаратних пристроїв, API і онлайн-сервісів новими і цікавими способами.

Node-RED надає редактор потоків на основі браузера, який дозволяє легко об'єднувати потоки, використовуючи широкий діапазон вузлів в палітрі. Потім потоки можна розгорнути під час виконання одним клацанням миші.

Функції JavaScript можна створювати в редакторі за допомогою редактора тексту фіксованої. Вбудована бібліотека дозволяє зберігати корисні функції, шаблони або потоки для повторного використання. Полегшена Виконавча побудована на Node.js і в повній мірі використовує його неблокіруючіх модель, керовану подіями. Це робить його ідеальним для роботи на кордоні мережі на недорогому обладнанні, такому як Raspberry Pi, а також в хмарі.

Маючи більш 225 000 модулів в репозиторії пакетів Node, легко розширити діапазон вузлів палітри для додавання нових можливостей. Потоки, створені в Node-RED, зберігаються з використанням JSON, який можна легко імпортувати і експортувати для спільного використання з іншими (рис.2.27).

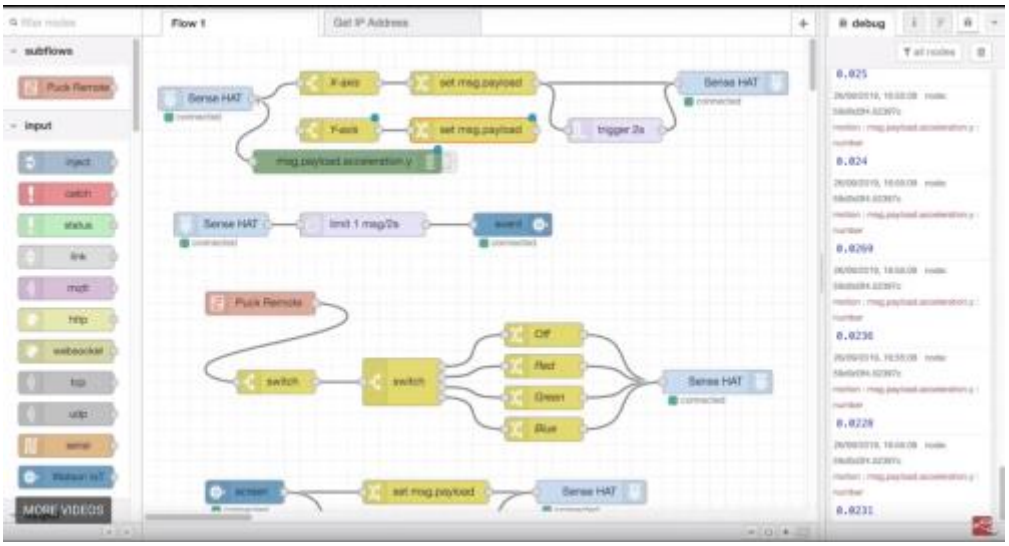


Рис. 2.27 Платформа Node-RED − інструмент програмування IoT для об'єднання апаратних пристрої

2.3 Вибір систем для тестування покращеної архітектури

На основі опрацьованих систем управління та автоматизації для розумних будинків зрозуміла, що дана задача досить популярна у користувачів і таке різноманіття варіантів свідчить про те, що на сьогоднішній день не існує ідеального рішення і кожен намагається знайти підходящу для своїх задач платформу. Саме тому важливим фактором у проектуванні архітектури пристроїв являється максимально універсальний підхід і абстрагування від системи управління. Таке рішення дозволить не зковувати користувача в рамках однієї платформи, а дасть йому можливість вибору більш підходящої.

Таким чином опрацювавши велику кількість платформ було зрозуміло, що кожна із них має свої переваги та недоліки, кожна дозволяє вирішувати багато різноманітних задач та має свої переваги.

Для тестування розроблених пристроїв і їх програмного забезпечення було обрано декілька платформ по декількох простих критеріях:

- доступність платформи
- простота в налаштуванні
- мінімальний час для підключення пристроїв з боку платформи
- максимальне різноманіття корисного функціоналу, що підтримується відразу після налаштування

Цим критеріям повністю відповідає система HomeBridge, оскільки не орієнтована на велику кількість інтеграції і створена для вирішення декількох задач, а саме швидке налаштування та запуск, швидке підключення девайсів, можливість керування з веб інтерфейсу та налаштування голосових асистентів. Саме дві останні можливості показують, що при правильно побудованій архітектурі на стороні пристроїв розумного дому, їх можна підключати в системи, які дають інструмент для вирішення нетривіальних задач, таких як голосове керування з мобільного телефону, що являється не простою задачею і змушує окремих виробників вкладати в їх вирішення з боку систем автоматизації.

Висновки

В другому розділі роботи було виконано огляд сучасних підходів до управління система розумного дому. На основі аналізу було виділено два основних способа:

- закрите програмне забезпечення від виробників розумних девайсів
- глобальні платформи для створення єдиної системі управління с відкритим вихідним кодом

На основі огляду можна зробити висновок, що перший спосіб унеможливлює, або робить інтеграцію сторонніх пристроїв дуже складною і не дає достатньої гнучкості для кінцевого користувача виконувати поставлені задачі по автоматизації власного будинку або промисловості. Другий же спосіб, дає можливість користувачам пристроїв розумного дому вибір в керуючій платформі, дає можливість обрати систему, що може задовольнити основні потреби та виконати поставленні задачі, крім цього через великі скупчення цільової аудиторії полегшить процес налаштування та зробити його максимально комфортним.

Після проведено огляду існуючих способів та підходів по управлінню, було визначено основний вектор розвитку власної архітектури, а саме пошук оптимального способу інтеграції з більшістю сучасних платформ автоматизації задля найкращого користувальницького досвіду користувача.

Для інтеграції в існуючі систему існує декілька основних способів:

- інтеграція за допомогою існуючого API
- інтеграція за допомогою створених модулів чи розширень
- створення універсальних рішень, що дозволить використовувати відкриті API чи розширення в уже готових екосистемах автоматизації

РОЗДІЛ 3
ВИБІР ІНСТРУМЕНТІВ І АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УДОСКОНАЛЕННЯ АРХІТЕКТУРИ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ

На сьогоднішній день найскладнішим завдання для користувача є не тільки питання управління своїми пристроями розумного дому, а ще й доступність їх інтеграції в сучасні системи управління. Це дозволить зручно та в єдиному підході налаштовувати процес управління всією екосистемою з однією точки.

Крім цього, важливою задачею є підтримка програмного забезпечення в актуальному стані та простота його модифікації в разі потреби.

Для вирішення цих та інших важливих питань,що покращують користувальницький досвід, на етапі планування архітектури було поставлено ряд пріорітетних задач, які мають стати основними продуктовими перевагами та покращать існуючі архітектури та підходи в процесах управління розумним домом.

3.1 Визначення основних функціональних та нефункціональних вимог до архітектури програмного забезпечення пристроїв

Опрацювавши ряд сучасних систем управління розумним домом було сформовано розуміння та підходи до роботи з девайсами. На основі цього розуміння було створено перелік функціональних та нефункціональних вимог для розроблюваної архітектури.

Функціональні вимогу до програмного запезпечення пристрою:

1. Встановлення і оновлення програмного забезпечення на пристрої повинно виконуватись універсально, згідно уніфікованого інструменту (CLI чи веб інтерфейсу)

2. Повинна бути можливість оновлювати програмне забезпечення за допомогою безпровідного інструменту (OTA update) для підвищення користувальницького досвіду

3. Архітектура має дозволяти інтегрувати пристрій в більшість сучасних систем управління розумним домом за допомогою уніфікованого підходу

4. За допомогою програмного забезпечення має бути можливість керувати більшістю існуючих прикладних пристроїв та считувати телеметрію основних датчиків та сенсорів

5. Підхід по управлінню пристроєм та зчитуванню телеметрії має бути уніфікованим в рамках компонентів

Нефункціональні вимоги до архітектури програмного забезпечення:

1. Архітектура має легко масштабуватись для підключення нових модулів підтримки нових датчиків та пристроїв

2. Керування пристроєм має бути можливим в рамках локальної мережі або через хварний веб сервер

3. Процес встановлення програмного забезпечення має бути можливим на більшості сучасних операційних системах

4. Для встановлення та оновлення ПЗ не має вимагати встановлення великої кількості сторонніх програм та інструментів

5. Пристрій має корректно обробляти випадки вимкнення енергозабезпечення та перебої в роботі локальної мережі

3.2 Визначення способу інтеграції з існуючими системами управління

Опрацювавши більше 25 сучасних платформ було зроблено висновок по можливим способам інтеграції з ними.

Найбільш популярним способом, який пропонують самі виробники платформ автоматизації являється розробка власних модулів чи розширення на рівні архітектурі програмного забезпечення (рис. 3.1). Такий підхід має ряд переваг - стабільність роботи через точний опис вимог зі сторони платформи,

можливість використання ресурсів платформи для розробки та інші. Але недоліків у такого способу значно більше:

- розробка підтримки модуля конкретної платформи не дає кінцевому користувачу у виборі, а тому унеможливлює інтеграцію з різними платформами і не виконує одного із поставлених функціональних вимог
- сковує використання пристрою в рамках однієї екосистема, яка може не підходити всім користувачам
- потребує завчасного оновлення ПЗ в випадку зміни роботи модуля зі сторони платформи
- потребу в вивченні підходів до розробки крім інтеграції, ще і розробки самих модулів на самій платформі (це можуть бути як сторонні мови програмування так і окремі підходи до розробки, які диктує платформа)

Оскільки даний підхід не дає можливості виконати всі поставлені вимоги до архітектури, його використання стає неможливим.

Другим по популярності способом інтеграції являється розробка підтримки відкритих API платформ на стороні ПЗ (рис. 3.2). Опрацювавши всі знайдені платформи було зроблено висновок, що більшість з них мають відкрите API для інтеграції, але в кожній системі структура і підходи роботи з ними повністю відрізняються. Цей факт свідчить про те, що даний спосіб так само як і перший унеможливлює виконання вимоги до універсальності. Хоча цей спосіб і не змушує розробляти ПЗ на рівні платформи, але має всі інші недоліки, що і перший.

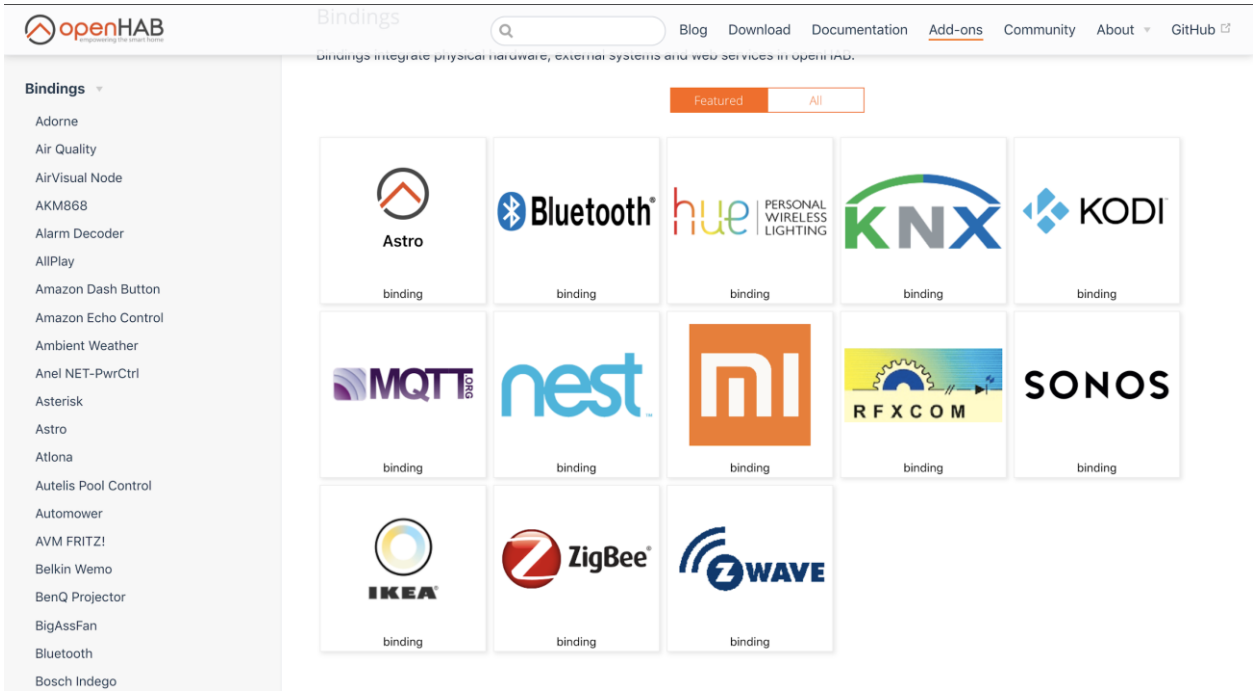


Рис. 3.1 Існуючі варіанти модулів в платформі OpenHab

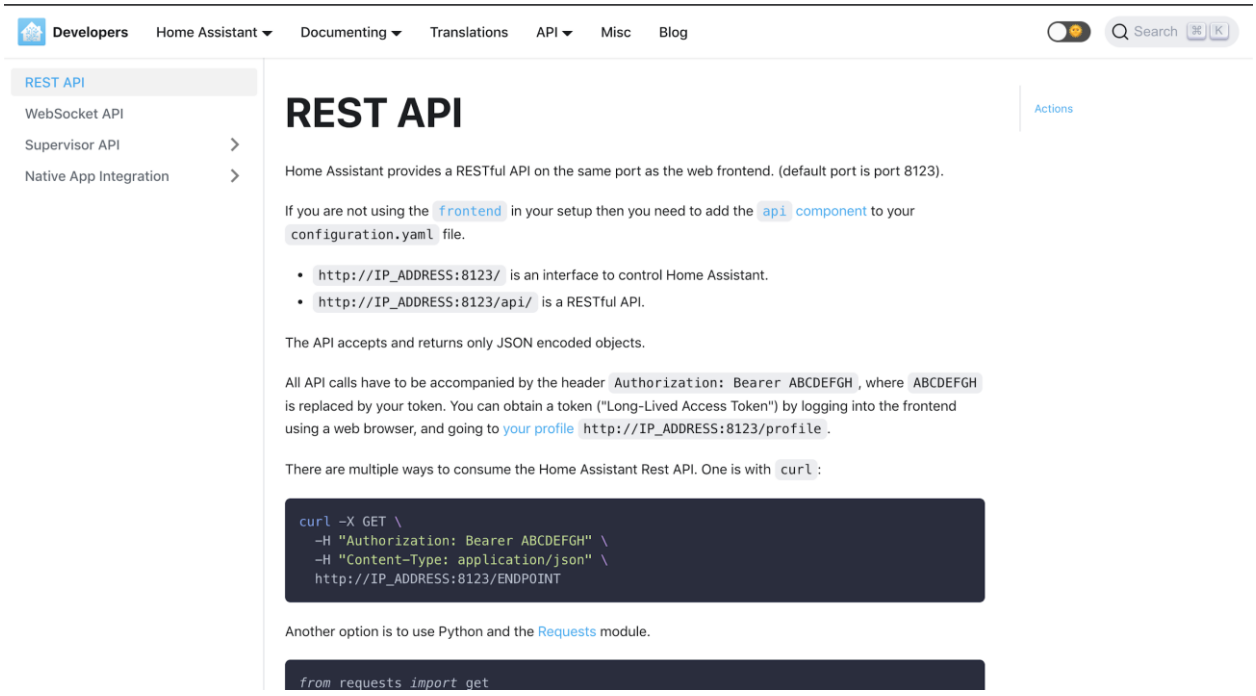


Рис. 3.2 Документація відкритого API Home Assistant

Третім способом для інтеграції являється використання можливостей уже існуючих модулів чи компонентів систем управління, виконаних

стороннім розробниками. Даний спосіб являється досить популярним саме через те, що в більшості платформ значно розвинута спілка користувачів, окрему нішу яких займають розробники, які беруть активну участь у розвитку проекту та займаються розробкою зовнішніх модулів для своїх потреб чи потреб інших користувачів.

Виконуючи аналіз сучасних платформ було звернено увагу на те, що одним із популярних модулів являється модуль MQTT брокеру. Це технологія досить розповсюджена в сфері IoT і являється одним із основних способів інтеграції пристроїв. Доказом цього факту являється те, що у 22 з 26 опрацьованих платформ MQTT модуль є доступним для встановлення, а в деяких платформах входить до складу ядра всієї платформи, що ще більше доводить його працездатність та розповсюдження.

Так наприклад, в OpenHab компонент MQTT являється найбільш популярним розширення всієї платформи.

На основі аналізу було прийнято рішення використовувати саме третій підхід, а саме виконувати інтеграцію с сучасними платформами через вбудовані MQTT модулі. Такий підхід дозволяє уніфіковано інтегруватись с платформами, що підтримують MQTT, а таких більшість. Крім цього більшість логіки інтеграції буде виконуватись на рівні готових інструментів самих компонентів,а архітектура ПЗ пристрою розумного дому буде абстрактно працювати по стандартній технології через MQTT брокер.

3.3 Вибір універсальної самодекларативної конвенції для опису стану девайсу

MQTT - це технологія, яка спочатку використовувалася для створення з'єднань в супутникової мережі. Легкий протокол дозволив знизити пропускну здатність і енергоспоживання.

Ефективність використання ресурсів MQTT зіграла важливу роль в системі, працюючи через дорогу супутниковий зв'язок. Пізніше MQTT став

використовуватися з більш доступними і недорогими каналами зв'язку і в різних областях застосування, включаючи Інтернет речей (IoT).

Принцип роботи

Протокол використовує шаблон публікації-підписки і включає в себе наступні компоненти:

Сервер або брокер, який спілкується з клієнтами (видавцями і передплатниками) через інтернет-з'єднання або локальну мережу.

Публікатор створює повідомлення і публікує їх з певної теми [24].

Підписник отримує повідомлення, що відносяться до теми, на яку він підписаний (рис. 3.3).

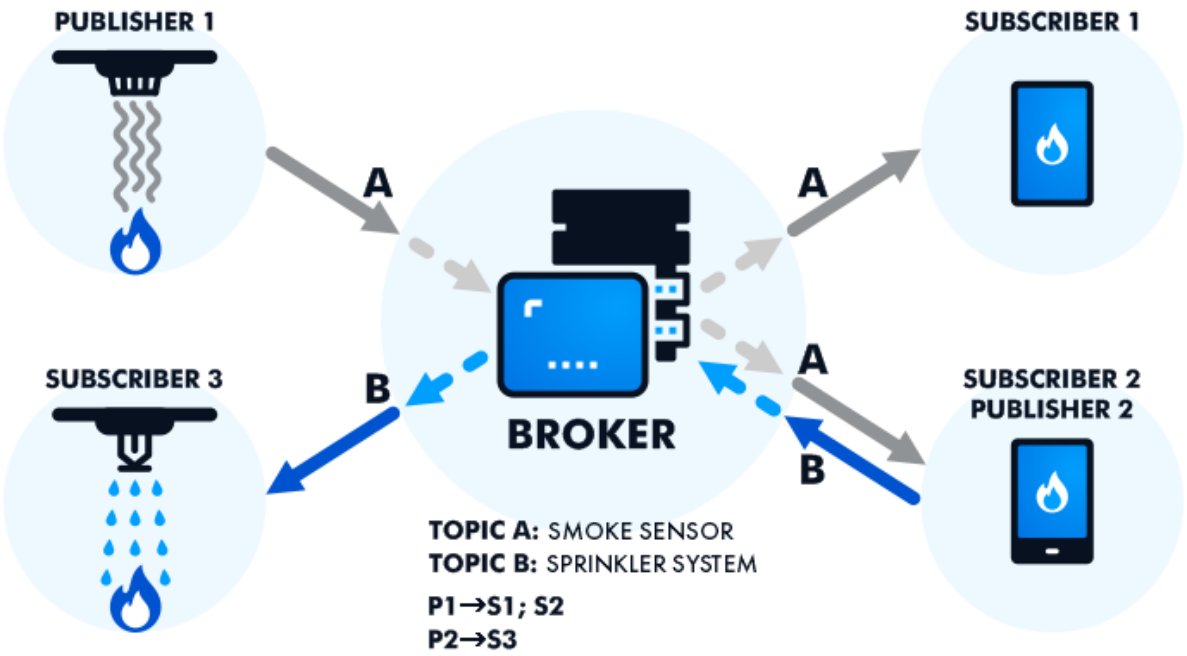


Рис. 3.3 Схема роботи MQTT

Публікатори і підписники можуть помінятися ролями, а іноді клієнти MQTT беруть на себе обидві ролі. Кількість видавців і передплатників може бути необмеженим. Це буде залежати тільки від потужності вашого сервера. У клієнтів є ідентифікатори, за якими їх ідентифікує сервер.

Кожна тема складається з різних підтем або рівнів, що утворюють ієрархію. Публікуючи повідомлення для конкретних підтем, видавець зможе зв'язатися з передбачуваним одержувачем, у якого є підписка на ті ж рівні тем.

Як і будь-який інший двійковий протокол, MQTT має перевагу перед текстовими протоколами в міжмашинної взаємодії. Пристрої можуть відправляти і отримувати виконавчі дані без додаткової обробки, що прискорює обмін повідомленнями в мережі.

MQTT передає повідомлення пакетами. Пакет складається з:

- Фіксований заголовок - це обов'язкова частина повідомлення, що містить контрольний заголовок і розмір пакета. Мінімальний розмір становить 2 байта, а максимальний - 5 байтів.

- Тема змінної - це необов'язкова частина повідомлення, що містить додаткову інформацію. Його розмір може варіюватися в залежності від типу повідомлення.

- Корисне навантаження - це необов'язкова частина повідомлення з максимальним розміром 256 МБ. Він може включати в себе різні команди, такі як включення / вимикання, обмін даними і читання даних датчиків (рис. 3.4).

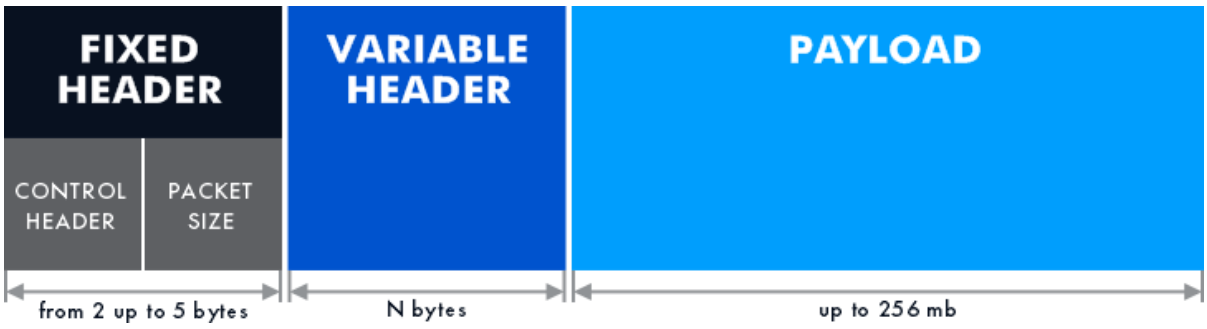


Рис. 3.4 Структура MQTT повідомлення

Залежно від можливостей ЦП MQTT може підключати багато тисяч або навіть мільйони пристроїв. Крім масштабованості, він простий у використанні. Він має низькі системні вимоги і високу сумісність з додатками, підключеними до Інтернету [25].

Ось чому ця мережева технологія так популярна в IoT. Він також знаходить широке застосування в побутовій електроніці, автомобілебудуванні та промисловості.

Використання в IoT

Протокол MQTT IoT може передавати дані навіть при нестабільних з'єднаннях. Він надає три варіанти якості обслуговування (QoS), які відповідають за доставку повідомлень. Вибір варіанту залежить від важливості даних і стабільності з'єднання:

QoS 0 (за замовчуванням): видавець відправляє повідомлення, не запрошуючи гарантовану доставку. Ви можете використовувати його, коли передається інформація не критична, а з'єднання стабільне.

QoS 1: видавець відправляє повідомлення, поки не отримає підтвердження доставки. Ви можете використовувати його, коли передається інформація критична, а з'єднання нестабільно. QoS 1 гарантує, що передплатник отримує повідомлення.

QoS 2: видавець відправляє повідомлення тільки один раз з гарантованою доставкою. Ви можете використовувати його, коли передається інформація критична, а з'єднання нестабільно. QoS 2 гарантує, що передплатник отримає повідомлення тільки один раз, без дублікатів та накладних витрат.

MQTT добре вписується в цю концепцію, що IoT це мережа підключених пристроїв, які обмінюються даними один з одним. Це легкий протокол з швидким часом відгуку, що робить взаємодію між пристроями ефективним, незалежно від їх кількості.

Привабливою особливістю архітектури протоколу MQTT є мінімальні накладні витрати. Це забезпечує плавну передачу даних з низькою пропускною здатністю і знижує навантаження на ЦП і оперативну пам'ять.

Бібліотеки MQTT з відкритим вихідним кодом і публічні брокери мінімізують витрати на розробку і прискорюють процес розробки.

Вільно доступна документація та велика спільнота розробників роблять реалізацію протоколу MQTT простий і можливої в будь-якому середовищі з використанням готових або призначених для користувача бібліотек.

Щоб розгорнути протокол MQTT для пристроїв IoT, вам знадобиться бібліотека і брокер, який може бути вашим локальним або хмарним сервером.

Безпека

Якщо ви плануєте використовувати MQTT в своєму додатку IoT, вам необхідно переконатися, що передані дані є конфіденційними і безпечними.

Безпека протоколу MQTT заснована на криптографічних протоколах TLS / SSL і досить слабкою аутентифікації. Так що MQTT не входить в число кращих протоколів з точки зору безпеки. Ви можете ввести додаткові функції, щоб забезпечити надійний захист MQTT і захистити свою систему IoT від вразливостей.

Ви можете додавати механізми безпеки на різних рівнях. Наприклад, ви можете використовувати брандмауер для захисту брокера на мережевому рівні. Або ви можете використовувати алгоритми шифрування для забезпечення безпеки MQTT на рівні додатку. Впровадження наскрізного шифрування дозволяє передавати конфіденційні повідомлення між клієнтами.

Ви також можете використовувати токен доступу для ідентифікації клієнтів та запобігання несанкціонованого доступу до всієї системи.

Щоб поліпшити аутентифікацію, вам потрібно змінити протокол і, отже, брокера. Оскільки стандартний брокер не допускає таких змін, вам доведеться розгорнути свого власного брокера і налаштувати його [26].

IoT додатки, що використовують MQTT

MQTT - це гнучка і проста у використанні технологія, яка забезпечує ефективний зв'язок в системі IoT. Так що не дивно, що існує безліч популярних додатків Інтернету речей, які використовують протокол MQTT. Наприклад, платформа IBM Watson IoT Platform використовує MQTT в якості основного протоколу зв'язку.

Amazon Web Services використовував MQTT в якості основи для сервісів AWS IoT. AWS IoT Core має брокер повідомлень на основі MQTT і підтримує два рівня MQTT QoS.

Microsoft Azure пропонує службу Azure IoT Hub для створення додатків IoT з використанням протоколу MQTT.McAfee, Red Hat, Cisco і IECC також входять в число компаній, які обрали MQTT для вирішення різних завдань.

Інші протоколи зв'язку, такі як HTTP, CoAP, AMQP, мають аналогічну функціональність. Ви також можете використовувати їх в додатках Інтернету речей. Щоб отримати точне уявлення про продуктивність і значущості протоколу MQTT для Інтернету речей, давайте розглянемо ситуацію в перспективі [27].

Альтернативи протоколу та їх порівняння

Безперебійне спілкування - один з ключових моментів добре функціонуючої системи Інтернету речей. Ви можете домогтися цього, вибравши відповідний протокол зв'язку. MQTT може стати хорошим варіантом для вашого IoT-продукту, але є і альтернативи.

HTTP та MQTT

HTTP використовує TCP / IP в якості транспортних протоколів, але майбутній HTTP / 3 буде заснований на UDP.

HTTP - це веб-протокол, який часто використовується разом з HTML для зв'язку з веб-браузерами.

HTTP - це текстовий протокол, який працює по моделі запиту / відповіді і використовує URI або URL в якості тем повідомлень. Клієнт відправляє запит на сервер, а сервер відправляє відповідь. Немає ніяких конкретних обмежень на розмір повідомлення; це сервер, який встановлює обмеження.

На відміну від MQTT, HTTP є великоваговим, тому що це текстовий протокол. Це має на увазі передачу повідомлень великого розміру і великі накладні витрати.

HTTP важкий. Йому потрібно більше енергії і ресурсів пам'яті, ніж MQTT і будь-якого іншого легковаге протоколу.

HTTP можна використовувати з кількома пристроями. Однак чим більше пристроїв ви додаєте, тим більше навантаження ви надаєте на сервер. Це негативно позначається на загальній продуктивності.

HTTP не має якості обслуговування або інших служб, тому покладається на TCP тільки як на гарантію доставки повідомлень.

Спочатку HTTP був розроблений без механізмів безпеки. HTTPS - це зашифрований HTTP, який має той же рівень безпеки, що і MQTT. Він використовує TLS / SSL для аутентифікації клієнтів і шифрування переданих даних [28].

CoAP та MQTT

Протокол обмеженого застосування (CoAP), розроблений IETF, є протоколом на основі UDP. У нього багато спільного з HTTP, і ці два протоколи надійно працюють один з одним.

CoAP - це бінарний протокол, який підтримує два режими зв'язку: публікація / підписка і запит / відповідь. Як і HTTP, він публікує повідомлення по URI. Максимальний розмір повідомлення визначається потужністю сервера.

Як і призначене для пристроїв з обмеженнями, CoAP має невеликий розмір повідомлення і мінімальні накладні витрати. Крім того, CoAP використовує UDP - простий транспортний протокол, який не має підтверджень зв'язку та гарантованої доставки. Це також мінімізує накладні витрати і розмір повідомлення. В цьому випадку CoAP має перевагу над MQTT. Незважаючи на легку архітектуру і невеликий розмір повідомлення, MQTT працює по протоколу TCP, який містить запити на з'єднання, рукостискання і гарантію доставки.

CoAP розроблений для пристроїв з обмеженими ресурсами. Так що його основні переваги - це енергоефективність і низьке споживання пам'яті. Іноді CoAP може бути більш ресурсоефективність, ніж MQTT.

AMQP та MQTT

Розширений протокол черги повідомлень (AMQP) був розроблений в JPMorgan Chase & Co. в 2003 році як відкритий протокол обміну повідомленнями для банківських систем.

AMQP - це двійковий протокол на основі TCP, який може використовувати або модель публікації / підписки, або шаблон запиту / відповіді.

Компонент обміну протоколу отримує повідомлення від видавця і направляє його в чергу. Черга зберігає повідомлення до тих пір, поки клієнт не зможе його обробити.

AMQP легкий і не вимагає великих витрат. Але він приділяє велику увагу доставку повідомлень, що збільшує накладні витрати і робить їх більш високими в порівнянні з MQTT.

AMQP не вимагає занадто багато ресурсів оперативної пам'яті і процесора. Однак він споживає їх більше, ніж MQTT, через більш складної системи обміну повідомленнями.

AMQP масштабований і дозволяє використовувати декілька підключень.

Обидва протоколу, AMQP і MQTT, мають функції і можливості для забезпечення надійного обміну повідомленнями. Вони працюють по TCP і застосовують QoS для забезпечення стабільної та надійної передачі даних. AMQP забезпечує два рівня QoS, а MQTT - три з них. Крім того, в MQTT є так звана опція «останньої волі і заповіту». Гарантує доставку повідомлення клієнту в разі відключення.

AMQP має перевагу перед MQTT з точки зору безпеки. Він використовує різні розширення TLS, включаючи SNI і SASL, для аутентифікації клієнтів і шифрування даних [29].

Бенчмаркінг пропускної здатності і затримки

Щоб отримати більш чітке уявлення про продуктивність MQTT в порівнянні з його альтернативами, ми провели власний порівняльний аналіз. Ми створили середовище тестування для вимірювання затримки і пропускної здатності. Це вимірні параметри, які підкреслюють ефективність протоколу.

Ми визначили пропускну здатність як максимальну кількість повідомлень в секунду, що відправляються від видавця брокеру або сервера. При вимірюванні затримки ми взяли мінімальний час відгуку сервера. Ми використовували Apache JMeter і його плагіни як інструмент навантажувального тестування.

Устаткування видавця працювало на одноядерному процесорі з робочою частотою 700 МГц і 512 МБ оперативної пам'яті. У серверному обладнанні використовувався одноядерний процесор з 2 ГБ оперативної пам'яті. Розмір корисного навантаження кожного повідомлення, яке ми відправляли з використанням протоколів, становив 64 КБ.

Згідно з нашими результатами тестування (рис. 3.5), CoAP показав видатну продуктивність по пропускній здатності. Йому вдалося відправити вдвічі більше повідомлень, ніж MQTT з QoS 0. HTTP і MQTT з QoS 1 і QoS 2 були приблизно на одному рівні з найнижчою пропускною спроможністю.

Ті ж протоколи (HTTP; MQTT з QoS 1 і 2) мали затримки в часі відповіді сервера. MQTT з QoS 0, CoAP і AMQP показали нульову затримку.

PARAMETER	MQTT (QoS 0)	MQTT (QoS 1)	MQTT (QoS 2)	AMQP	CoAP	HTTP
THROUGHPUT (msg/sec)	75,5	14,3	7,8	67	148,3	11,3
LATENCY (ms)	0	58	56	0	0	18,2

Рис. 3.5 Результати бенчмарків

Перспективи протоколу

Згідно Google Trends (рис. 3.6), за останні 5 років рівень інтересу до MQTT у всьому світі виріс в чотири рази. Ця тенденція підтримується швидким розширенням ринку Інтернету речей і стійкою тенденцією до простоти і ефективності використання ресурсів.

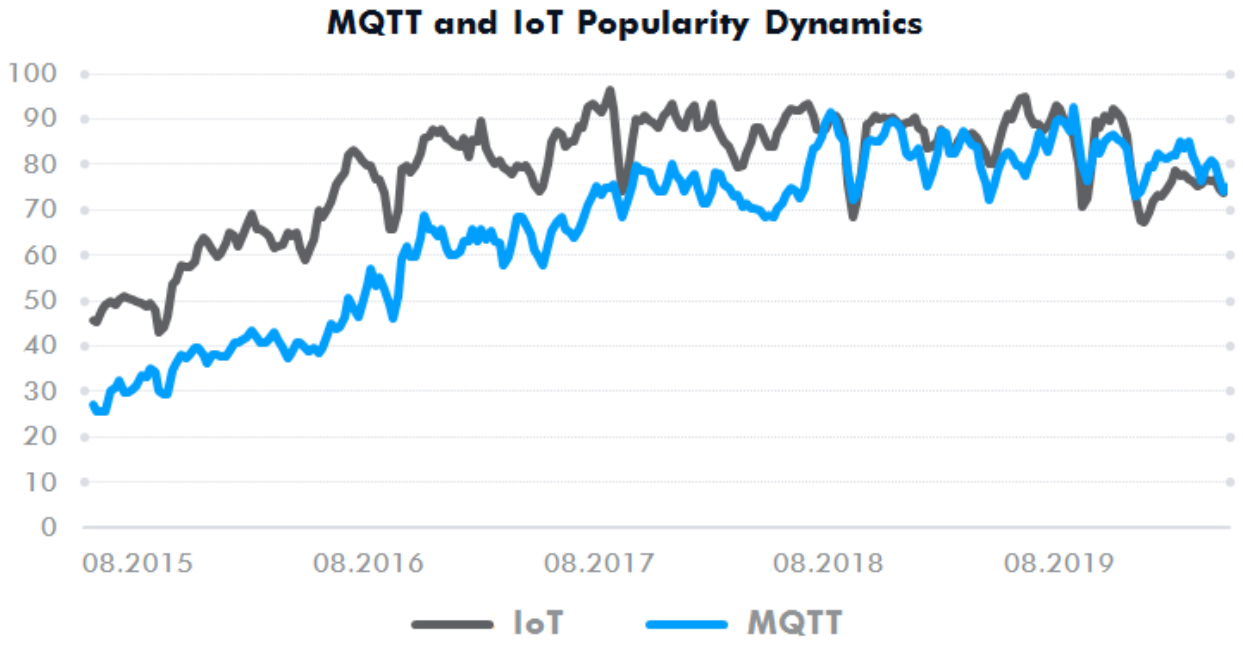


Рис. 3.6 Графік популярності протоколу

Проста архітектура, легка реалізація, надійна доставка повідомлень і висока масштабованість - сильні переваги, які збережуть інтерес до протоколу MQTT для IoT.

OASIS випустив останню версію протоколу, MQTT 5.0, в березні 2019 року. У v5.0 додані нові функції: закінчення терміну дії повідомлення, загальна підписка і псевдонім теми. Вони оптимізували продуктивність MQTT і зробили його більш привабливим для розробників Інтернету речей [30].

Модель публікації / підписки

На відміну від традиційної моделі клієнт-сервер, в якій клієнт безпосередньо пов'язується з кінцевою точкою, клієнти MQTT розділені на дві групи: відправник (званий в MQTT видавцем) і споживач, який отримує дані

(передплатник MQTT). Видавець і передплатник нічого не знають один про одного і, по суті, ніколи не знаходяться в прямому контакті один з одним. Третій компонент (брокер MQTT) діє як «даішник», направляючи повідомлення від видавця до будь-яких кінцевих точок, що виступає в якості передплатників (рис. 3.7).

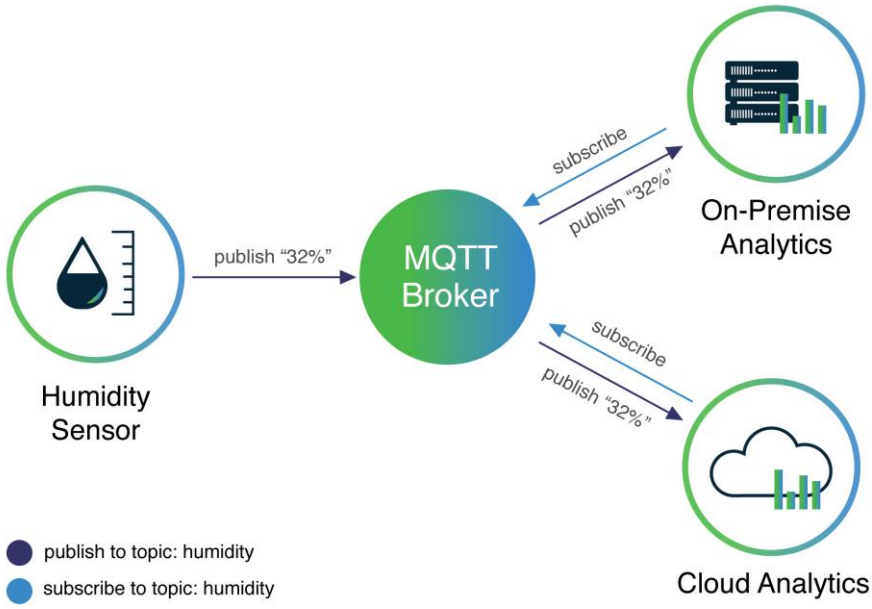


Рис. 3.7 Модель публікації-підписки

Теми MQTT

Повідомлення відправляються від видавця через брокера до одного або декількох передплатникам, які використовують теми. Теми являють собою ієрархічні рядки UTF-8. Кожен рівень в темі відділяється косою рисою. Кожне повідомлення від видавця має включати тему. Щоб отримати опубліковане повідомлення, об'єкт, який використовує повідомлення, повинен підписатися на ту ж тему. Брокер відправляє отримане повідомлення тільки тим клієнтам, які підписалися на ту ж тему.

Отже, MQTT - сильний конкурент в області передачі даних і мережевих комунікацій, хоча він не є номером один у всіх відносинах. Наприклад, CoAP

перевершує MQTT по пропускній здатності і накладних витрат. І AMQP грає провідну роль в забезпеченні безпеки.

Однак є кілька ключових переваг, які MQTT надає своїм користувачам:

- надійність;
- швидкий час відгуку;
- можливість підтримки необмеженої кількості пристроїв;
- публікувати / підписуватися на повідомлення, які ідеально підходять для спілкування "багато до багатьох";
- достатня документація і сильне співтовариство розробників, які спрощують реалізацію протоколу MQTT.

Все це робить MQTT підходящим варіантом для IoT-пристрої. При розробці рішення IoT MQTT використовується як частина стека протоколів. Це один з улюблених варіантів для малопотужних бездротових пристроїв з нестабільним підключенням.

Однак MQTT не визначає структуру і зміст цих повідомлень і їх взаємозв'язок. Пристрій IoT публікує дані і надає можливості взаємодії, але контролюючий об'єкт повинен бути спеціально налаштований, щоб мати можливість взаємодіяти з пристроєм. Перевага Homie визначає стандартизований спосіб того, як пристрої та онлайнових речей оголошують себе і свої дані на брокера MQTT.

Таким чином, це найважливіший аспект протоколу MQTT для автоматичного виявлення, настройки та використання пристроїв і служб [31].

3.4 Апаратне забезпечення для побудови власних пристроїв розумного будинку

Для проектування архітектури програмного забезпечення потрібно визначитись з апаратним забезпеченням, що буде виконувати написано ПЗ. На сьогоднішній день існує безліч різноманітних варіантів апаратного забезпечення на основі якого будуються пристрої розумного дому. Для роботи було виділено ряд основних вимог до апаратної частина для задоволення

потреб користувача і в той же час можливості виконання всіх функціональних та нефункціональних вимог до ПЗ.

Поставлені вимоги до апаратного забезпечення:

- доступність мікроконтролера на ринку
- відносна дешевизна мікроконтролера
- популярність
- доступність ресурсів для роботи з мікроконтролером
- мінімальна енерго потрібність
- варіативність складу мікроконтролера
- можливість модифікації

Плата мікроконтролера вважається невеликим комп'ютером, побудованим на мікросхемі з металооксидних напівпровідника. Всі типи мікроконтролерів складаються з одних і тих же основних частин корпусу:

- центрального процесора (ЦП)
- пам'яті
- периферійних пристроїв введення / виводу (програмовані).

Мікроконтролери використовуються в автоматично керованих продуктах і пристроях, таких як системи управління автомобільними двигунами, що імплантуються медичні пристрої, пульти дистанційного керування, офісні машини, побутова техніка, електроінструменти, іграшки та інші вбудовані системи.

Інженери, ентузіасти і програмісти також використовують одні і ті ж плати мікроконтролерів для створення проектів DIY і в навчальних цілях. В процесі аналізу було обрано 11 найпопулярніших на сьогоднішній день мікроконтролерів, які використовуються в розробці пристроїв розумного дому та хоча б в якійсь мірі відповідають поставленим вимогам до апаратного забезпечення.

Arduino Microcontroller Board

Компанія, що займається електронікою і технологіями, Arduino випустила версію Arduino Uno R3 в 2011 році. Вона заснована на чіпі під назвою ATmega328P (плата з відкритим вихідним кодом). Плата має різні контакти введення-виведення, за допомогою яких ви можете підключати її до інших платам і схемами. До нього підключені різні порти, включаючи порт USB-підключення, чотирнадцять контактів введення-виведення, заголовок ICSP, роз'єм джерела живлення і кнопку скидання. Його можна легко підключити безпосередньо до вашого персонального комп'ютера або ноутбука через USB-кабель.

Плата Arduino Uno R3, заснована на IoT, є найдешевшою, готової до підключення плати до різних доступним онлайн-бібліотекам і ресурсів (рис. 3.8). Крім цього, існує безліч інших комплектацій данного мікроконтролера з різними технічними характеристиками та вартістю.

Рис. 3.8 Arduino Uno R3

Teency 4.0

Плата мікроконтролера Teensy 4.0 (процесор 600 МГц) - новітня і найшвидша плата, доступна на сьогоднішній день. Вона має невеликі розміри в порівнянні з іншими дошками і може використовуватися для виконання різних типів проектів своїми руками. Всі команди подаються на плату через два порти USB. Teensy 4.0 можна запрограмувати за допомогою Arduino IDE за допомогою невеликого доповнення.

Мікроконтролер можна підключити до ПК або ноутбука за допомогою кабелю USB (рис. 3.9). Він має 1024 КБ ОЗУ в порівнянні з 16 КБ у Arduino Uno для більш сучасних додатків [32].

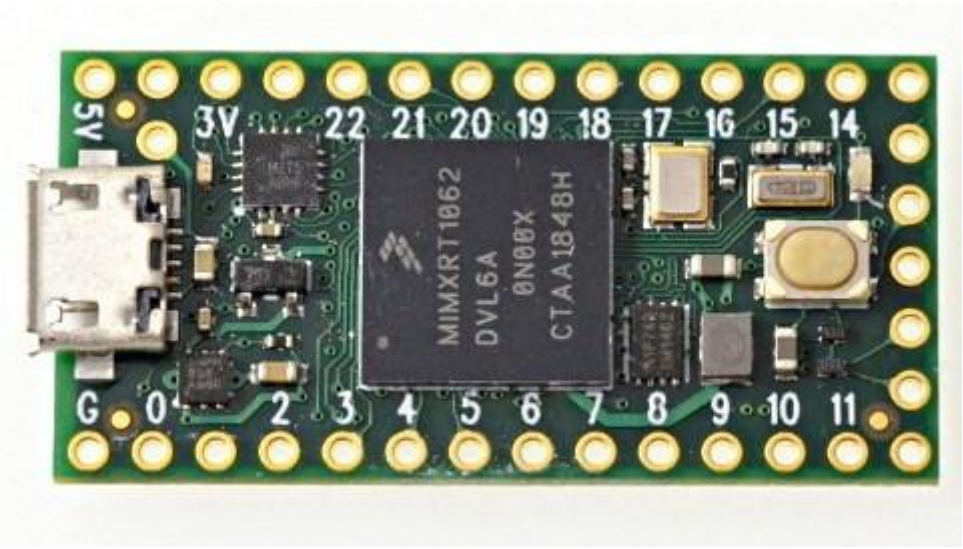


Рис. 3.9 Teensy 4.0

Arduino Pro Mini 328

Arduino Pro Mini 328 - ще одна найкраща плата мікроконтролера від технології Arduino. Ця міні-плата призначена тільки для невеликих додатків до 5 вольт. Плата Arduino Pro Mini 328 має завантажувач на 16 МГц. На платі немає вбудованих роз'ємів і портів, тому вам, можливо, доведеться припаяти з'єднання самостійно. Однак якщо у вас невеликий бюджет, то ця плата мікроконтролера - хороший вибір для вас (рис. 3.10).

Рис. 3.10 Arduino Pro Mini 328

Raspberry Pi 4

Випущена в 2019 році Raspberry Pi 4 є найшвидшою платою мікроконтролера на сьогоднішній день. Завдяки 4 ГБ оперативної пам'яті ви можете створювати потужні і просунуті електронні проекти. Raspberry Pi 4 може забезпечувати струм до 1,2 А для USB-пристроїв. Доступні різні варіанти оперативної пам'яті від 1 ГБ до 4 ГБ. Додаткові функції включають вбудовану бездротову локальну мережу, Bluetooth 5.0, два порти USB 2.0 і USB 3.0, два порти Micro HDMI і порт Gigabit Ethernet (рис. 3.11).

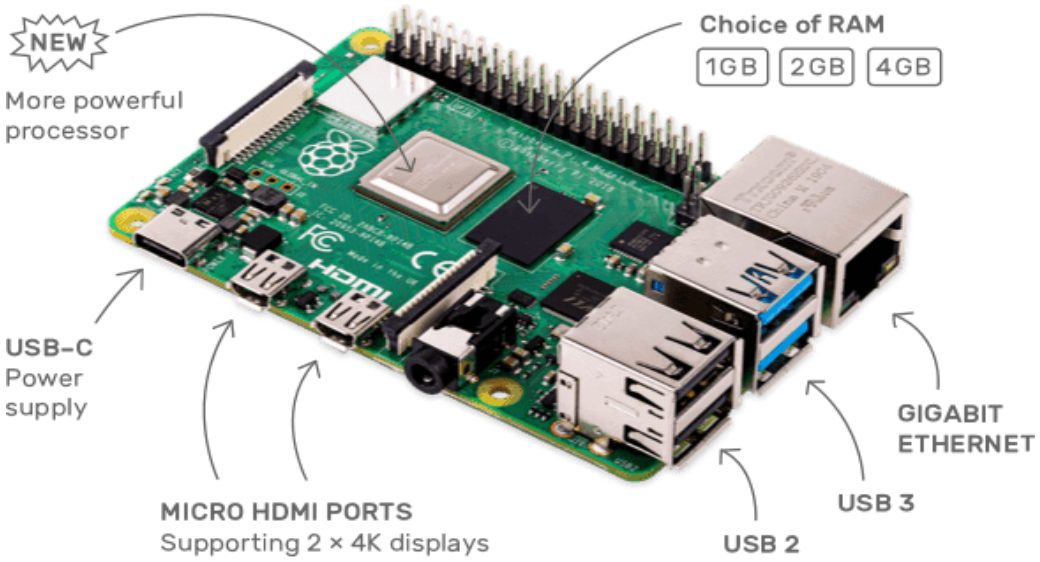


Рис. 3.11 Raspberry Pi 4

ESP32

Плата мікроконтролера ESP32 є комбінованим модуль Bluetooth і Wi-Fi на однокристальної платі (2,4 ГГц) з наднизьким енергоспоживанням. Плата вважається кращим вибором для додатків, де потрібно найкраща продуктивність RF. Плата мікроконтролера ESP32 використовується для проектів DIY, таких як розумний будинок і проекти на основі Інтернету речей (рис. 3.12) [33].

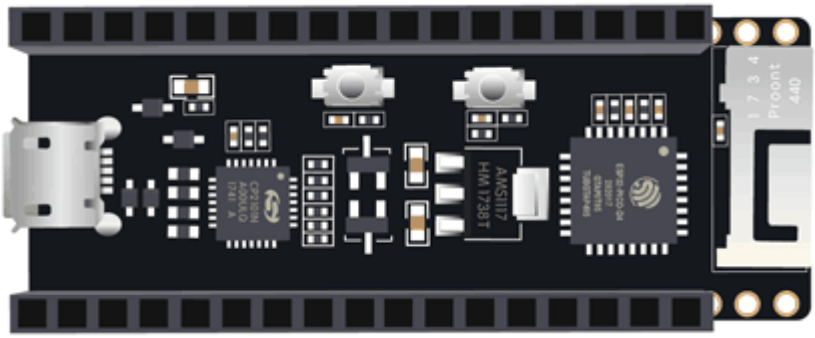


Рис. 3.12 ESP32

MBED LPC1768

Плата мікроконтролера MBED LPC1768 в основному призначена для створення прототипів. Він включає вбудований програматор USB FLASH. Плата заснована на 32-бітному ядрі ARM M3 NXP LPC1768. ОЗУ становить 32 КБ, флеш-пам'ять 512 КБ (рис. 3.13).

Він складається з периферійних пристроїв введення-виведення, порти USB і вбудованого Ethernet. Для нових розробників та інженерів у MBED є

кілька онлайн-спільнот для обміну бібліотеками та ресурсами, які можуть бути легко доступні будь-якому користувачеві для створення прототипів рішень.

Рис. 3.13 MBED LPC1768

BeagleBone Black

BeagleBone Black - одна з найдешевших доступних плат для розробки. Ви можете почати розробку всього за п'ять хвилин, підключивши свій комп'ютер за допомогою простого USB-кабелю. Він складається з 512 МБ ОЗУ і флеш-пам'яті 4G. Він має 46 × 2 контактів заголовка, Ethernet, 2 порти USB. Більша кількість контактів введення / виведення робить його більш відповідним для проектів електроніки. Він також має знижене енергоспоживання без потреби в радіаторах.

ESP8266

ESP8266 має невеликі розміри в порівнянні з іншими микроконтроллерами, що підтримують IoT. Він має дуже низьку вартість (близько 3 доларів США). Його можна використовувати для проектів розумного будинку своїми руками, включаючи Інтернет речей. Цю дошку також можна використовувати для створення ваших особистих помічників, таких як Кортан або Алекса. Він включає в себе 128 КБ ОЗУ і 4 МБ флеш-пам'яті, але що робить ESP8366 краще, так це те, що його можна використовувати для створення власної мережі для підключення інших пристроїв (рис. 3.15).

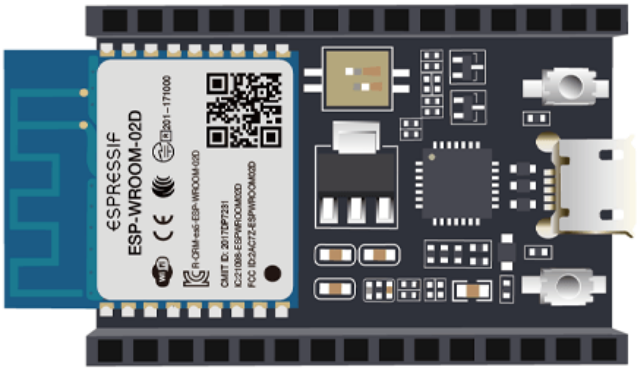


Рис. 3.15 ESP8266

Quark D2000

Мікроконтролер Quark D2000 є одним з найнадійніших мікроконтролерів і має більше елементів управління введенням-висновком, ніж інші мікроконтролери. Він заснований на сімействі мікроконтролерів Intel × 86. Це 32-бітний мікроконтролер, що працює на частоті 32 МГц з 8K SRAM і 32K FLASH. Він надзвичайно універсальний, оскільки вимагає постійного струму всього до 3,3 В (рис. 3.16).

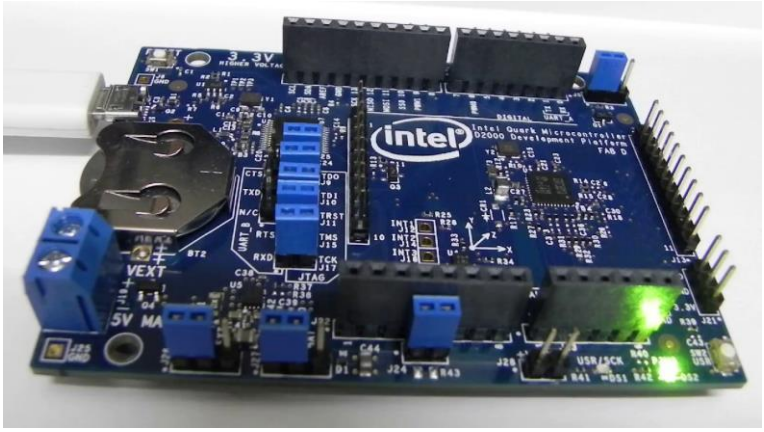


Рис. 3.16 Quark D2000

Launchpad MSP430

Launchpad MSP430 найбільш корисний для застосування вбудованої емуляції і налагодження Energy Trace. Це також мікроконтролер з наднизьким енергоспоживанням і 4 КБ ОЗУ, який можна використовувати в якості альтернативи для Arduino Uno R3. Він має спеціальне програмне забезпечення для програмування (IDE), яке використовується для кодування і налагодження, яке називається Energia, воно більш зручне для користувача і схоже на IDE Arduino (рис.3.17).

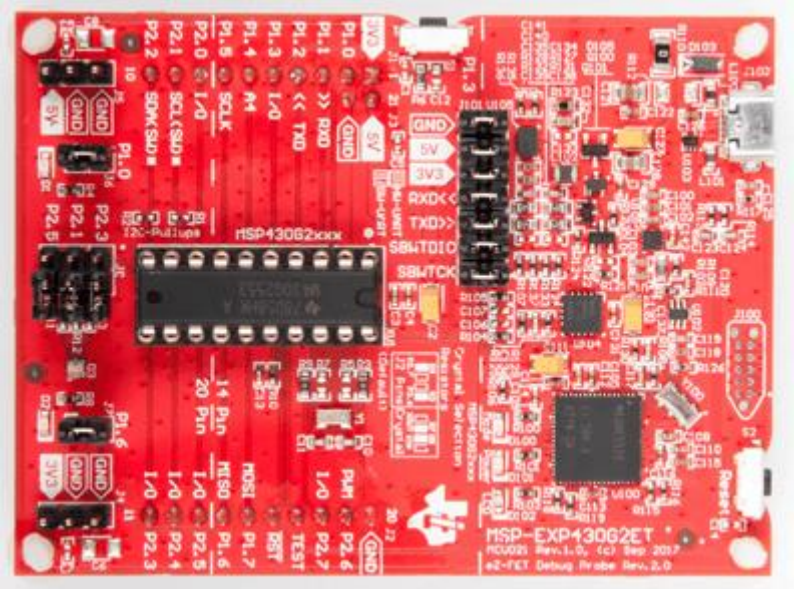


Рис. 3.17 Launchpad MSP430

Raspberry Pi Zero W

Raspberry Pi Zero W розширює сімейство мікроконтролерів Pi Zero і поставляється з доданою WLAN активне можливістю підключення по Bluetooth. Він має всі функції оригінального Pi Zero. Він має одноядерний процесор з тактовою частотою 1 ГГц і 512 МБ оперативної пам'яті. Raspberry Pi Zero W - відмінний вибір для створення власних проектів вбудованого Інтернету речей (IoT) (рис. 3.20). Це дуже економічно і коштує близько 10 доларів США [34].

Рис. 3.18 Raspberry Pi Zero W

На основі наведеного списку можна зробити висновок, що на сьогоднішній день існує безліч способів створювати пристрої розумного дому на різній апаратній частині. Всі з описаних плат для програмування чудово підходять для створення на їх основі різних девайсів розумного дому, але не всі задовольняють поставленні вимоги.

На основі проведеного аналізу було обрано мікроконтролери ESP по ряду причин, а саме:

- мала вартість самого мікроконтролера
- варіативність його моделей та технічних характеристик
- вбудований Wi-Fi модуль
- велика кількість поціновувачів
- доступність документації, інструкцій та приладів
- зручність у використанні
- невеликі габарити

Крім цих переваг, найважливішим є те що більша частина сучасних Wi-Fi пристроїв розумного дому побудована саме на основі мікроконтролерів ESP, що дасть можливість використовувати створене програмне забезпечення на готових пристроях без потреби створення своїх плат та корпусів.

3.5 Вибір інструментів для створення програмного забезпечення пристроїв розумного будинку

Оскільки в виді апаратного забезпечення було обрано пристрої сімейства ESP, то і інструменти програмування потрібно обирати відштовхуючись від архітектури і можливостей мікроконтролера. На сьогоднішній день існує велика кількість інструментів для програмування, але потрібно обрати найбільш доцільний і сумісний с ESP фреймворк, який дозволяє виконати всі поставлені вимоги та задачі.

Для інструменту на якому має створюватись програмне забезпечення була поставлена одна основна вимога — низький поріг входу користувачів і можливість розробки програмного забезпечення самостійно без необхідності вивчення специфічних мов програмування.

Задля задоволення поставленої вимоги було проведено огляд існуючих інструментів і визначено ряд їх переваг та недоліків.

Micro Python

MicroPython - одна з програмних платформ для мікроконтролерів ESP32 і ESP8266. Тут використовується мова програмування Python 3, який включає невеликий підмножина стандартної бібліотеки Python і оптимізований для роботи на мікроконтролерах ESP.

Плати ESP32 і ESP8266 - це мікроконтролери з підтримкою Wi-Fi, які запускають MicroPython на «голому залізі», надаючи вам низкоуровневую операційну систему Python, яку можна використовувати для управління всіма видами електронних проектів.

MicroPython має розширені функції, такі як інтерактивне запрошення, закриття, генератори, обробка виключень і багато іншого. Ця операційна система буде працювати всього з 256 КБ кодового простору і 16 КБ ОЗУ.

Circuit Python

Схема Python заснована на Python. Це легко використовувати з платами ESP32 і ESP8266. Якщо у вас вже є знання Python, ви можете легко застосувати їх до Circuit Python. Якщо у вас немає досвіду, почати дуже просто.

Circuit Python простий у використанні, тому що все, що вам потрібно, це плата ESP32 або ESP8266, USB-кабель і комп'ютер з USB-з'єднанням.

Mu - це простий редактор коду для Circuit Python, він працює з Windows, Mac і операційною системою Linux. Перевірте наступне посилання для редактора Mu.

Arduino

Arduino дуже потужний і простий у програмуванні. У нього є велике всесвітнє співтовариство, так що воно може ділитися нашими знаннями і отримувати відповіді від багатьох людей. Arduino поставляється з власним програмним забезпеченням. Це програмне забезпечення працює з операційними системами Windows, Linux і Mac. Ви можете програмувати онлайн. Якщо ви знаєте C і C ++, це полегшить вам програмування.

Проект Arduino стартував в 2003 році як програма для студентів Інституту дизайну взаємодії Ивреа в Ивреа, Італія, з метою надати недорогий і простий спосіб для новачків і професіоналів створювати пристрої, які взаємодіють з навколишнім середовищем за допомогою датчиків і виконавчих механізмів. Поширеними прикладами таких пристроїв, призначених для початківців любителів, є прості роботи, термостати і датчики руху.

Після установки Arduino IDE необхідно встановити плату для ESP 8266 і ESP32, щоб завантажити і перевірити коди.

NodeMCU

NodeMCU - це прошивка на основі Lua для ESP32 і ESP8266 WiFi SOC від Espressif, що використовує файлову систему SPIFFS на основі вбудованої флеш-пам'яті. Прошивка спочатку розроблялася як супутній проект популярних модулів розробки NodeMCU на базі ESP8266, але тепер проект

підтримується спільнотою, і тепер прошивку можна запустити на будь-якому модулі ESP.

Mongoose

Mongoose os - це середовище розробки прошивки для ESP32 і ESP8266. ОС Mongoose - це платформа для створення додатків для малопотужних мікроконтролерів, яка складається з наступних основних компонентів:

Інструмент mos. Надає можливості управління пристроєм і створення прошивки,

Набір інструментів для збірки. Це образ докера, який містить SDK виробника обладнання разом з вихідними кодами mongoose-os. Mos збирає прошивку, беручи файл mos.yml в поточному каталозі і викликаючи образ докера збірки віддалено або локально.Колекція готових додатків і бібліотек.

Espruino

Espruino - це кілька варіантів мікроконтролерних пристроїв, в яких прошитий вбудований інтерпретатор JavaScript. Espruino Pico - саме мініатюрне з них. Оригінальний інтерпретатор JavaScript, що використовується в Espruino, призначений для швидкої розробки на пристроях з обмеженими процесорними ресурсами. Є його версії для цілого переліку платформ, починаючи з ESP8266 і до Raspberry Pi.

Для Espruino існує велика кількість готових підвантажуваних модулів для найрізноманітнішого периферійного обладнання, сумісного з екосистемою Arduino та інструкцій по його підключенню [35].

Провівши ознайомлення з різноманіття сучасних інструментів для програмування, було зроблено вибір в перевагу Espruino по ряду причин:

- Espruino має найнижчий поріг входу для недосвідченних користувачів
- більшість існуючих бібліотек із найбільш популярного Arduino або вже перенесені в інфраструктуру або є така можливість
- Espruino дозволяє найшвидше отримувати працююче ПЗ на мікроконтроллері

- є безліч готовиг бібліотек під різні пристрої та датчик
- має зручний веб інтерфейс
- повністю сумісна з більшістю видів мікроконтролерів сімейства ESP

Висновки

В третьому розділі було проведено ряд аналітичних задач для продумування майбутньої архітектури ПЗ для пристроїв розумного дому. В результаті аналіз сучасних платформ та систем управління було сформовано перелік функціональних та нефункціональних вимог для програмного забезпечення на основі сучасних потреб користувачів та їх повсякденних задач.

Для простої і гнучкої інтеграції із сучасними платформами, що дасть можливість керувати і зчитувати дані з пристроїв було обрано використання MQTT як протоколу для спылкування с брокером. Такий спосіб дає можливість підключати пристрої в більшість сучасних систем управління, або в звичайний MQTT брокер і керувати пристроєм з сторонніх веб або мобільних клієнтів. Для стандартизації і самодекларативного опису девайсів було обрано Homie як базову MQTT конвенцію.

Крім цього в результаті аналізу було отримано розуміння переваг та недоліків протоколу і можливих сценаріїв його використання.

В результаті данного розділ також було оброблено перелік більшості сучасних мікроконтролерів та їх можливостей. Вибір зупинився на мікроконтролерах сімейства ESP, через їх доступність, варіативність, низьку вартість та необхідну технічну складову для вирішення поставлених задач.

Останнім етапом було проведено аналіз можливих інструментів для програмування на мікроконтролерах ESP і обрано Espruino для реалізації проекту.

Наприкінці третього розділу було повністю сформовано ряд вимог, необхідних для початку розробки власної архітектури та визначено всі

технічні можливості та бар’єри з якими можна було зіштовхнутись в процессі розробки.

РОЗДІЛ 4
ВАРІАНТ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З УДОСКОНАЛЕНОЮ АРХІТЕКТУРОЮ

Для створення універсального програмного забезпечення було обрано підхід, згідно якого програмне забезпечення має конфігуруватись в єдиному файлі по описаним інструкціям. Таким чином все що необхідно користувачу — це ознайомитись з документацією по написанні конфігураційного файлу, описати його згідно конвенції та вказати його при завантаженні програмного забезпечення на пристрій і ПЗ буде сформовано на основі даного файлу. Такий підхід дозволяє повністю зберегти гнучкість для користувача і дати йому можливість використовувати всі можливості написаних модулів для роботи с фізичним сенсорами та пристроями в будь якій їх комбінації.

Обов'язковою задачею являлось дати можливість експертним користувачам самостійно масштабувати та розвивати проект власними силами. Така можливість стала можливою через винесення бізнес логіки роботи з пристроями та компонентами приладу через ізольовані модулі, кожен з яких опціонально може бути внесений в конфігурацій файл.

4.1 Реалізація автоматичної збірки програмного забезпечення на основі конфігураційного файлу

Першочерговою задачею після налаштування середовища розробки стало питання роботи с конфігураційним файлом, в саме його розбиття та передачу необхідних даних компілятору для збору необхідних бінарних файлів, які далі мають завантажуватись на пристрій.

Для описання конфігураційного файлу було обрано популярний і зрозумілий формат json. В самому початку було створено формат файлу і розбито його на основні компоненти.

Умовно файл конфігурації було розбито на декілька вкладених об’єкта (рис. 4.1). Весь основний контент контет було винесено в об’єкт *extensions* для

майбутнього масштабування. Основний об’єкт складається с декількох додаткових:

- *modules* — масив із підключених модулів с бібліотеки; такий формат опису необхідний для оптимізації використання пам’яті при створенні бінарних файлів ПЗ, він дозволить завантажити тільки ті модулі, що точно використовуються в девайсі, а всі інші буду проігноровані

- *controller* — атрибут, який вказує який тип мікроконтролера буде використовуватись на пристрої; цей атрибут являється обов’язковим, так як відштовхуючись від буде запущено спеціалізований процесс збирання ПЗ під час завантаження

- *mapping* — масив з описом співвідношення конкретних виходів мікроконтролера до модуля, який буде з ним взаємодіяти (далі буде детально описано яким чином формується цей масив для кожного із існуючих модулів)

Рис. 4.1 Розбиття конфігураційного файлу

Другим основним об’єктом файлу конфігурації являється *deviceConfig*. Саме в цьому місці конфігураційного файлу виконується формування структури девайсу в брокері згідно конвенції Homie.

Згідно конвенції кожен девайс представлений у вигляді набору дочірніх сутностей — нод. Кожна нода має свій набір сенсорів, опцій та телеметрії. Кожне з цих понять являється абстрактним і служить лише для уніфікованого підходу до відображення девайса в брокері. Крім цього, згідно конвенції, у самого девайсу можуть бути опції та телеметрії, але сенсори — це атрибути

тільки нод. Саме на основі цього конфігу буде формуватись стейт девайсу та його топіки в MQTT (рис. 4.2). На рисунку показано приклад об’єкту *deviceConfig,* що описує девайс, який складається с декількох телеметрій, одна з яких айпі адреса пристрою, однієї стандартної опції, та декількох нод. На рисунку видно, що відхід до описання сенсорів, опцій і телеметрій і девайсів і нод абсолютно однаковий, що свідчить про універсальність та єдиний підхід. На рисунку також зображено, що кожна нода має ряд специфічних обов'язкових атрибутів:

- *id* — унікальний ідентифікатор ноди, згідно вимог конвенції

- *state* — статус ноди за замовчування, цей атрибут являється не функціональним, але був диктований конвенцією, згідно якої статус має бути одним із списку (ready, init, lost, disconnected) і описувати її стан

- *type* — опціональний атрибут ноди, який потім може оброблятись в системах управління в їх бізнес логіці

- *sensors* — масив з об'єктів сутностей сенсора, в якому може бути описана будь яка комбінація сенсорів; по аналогії в ноди можуть бути такі ж масиви з об’єктів для options і telemetry, а уніфікований підхід дозволяє легко змінювати місце розположення цих сутностей.

Аналогічним способом потрібно описувати стан сенсорів, опцій та телеметрії і розміщувати їх в потрібних місцях девайсу (в ноді чи в самому девайсі). На рисунку 4.3 представлено список обов'язкових атрибутів сенсорів:

- *id* — унікальний ідентифікатор сенсора, опції чи телеметрії, згідно вимог конвенції

- *datatype* — ідентифікатор типу даних сутності згідно конвенції; може бути одним із списку (string, integer, float, boolean, color, enum); цей атрибут призначений для використання в бізнес логіці системи управління і ідентифікує які дані можуть бути передані або прийняті в топік даного сенсора

- *retained* — атрибут, який описує якого типу повідомлення будут відправлятись в брокер для топіків цього сенсору

- *unit* — атрибут, що описує статичну одиницю вимірювання значення даного сенсору;

Рис. 4.2 Приклад об’єкта с описов стану пристрою згідно конвенції

Рис. 4.3 Приклад об’єкта с описом стану пристрою згідно конвенції

settable — атрибут, який описує можливість зміни стану в топіках цього сенсору (так наприклад для датчика температури цей атрибут має бути false, оскільки відредагувати значення температури може лише сам мікроконтролер, а для аналогового виходу — true, оскільки сенсори такого типу будуть регулюватись користувачем)

- *name* — атрибут, що описує статичну назву сенсору; валідація цього атрибуту відрізняється від id, що дає можливість описати в ньому комфортнішу назву для ідентифікації

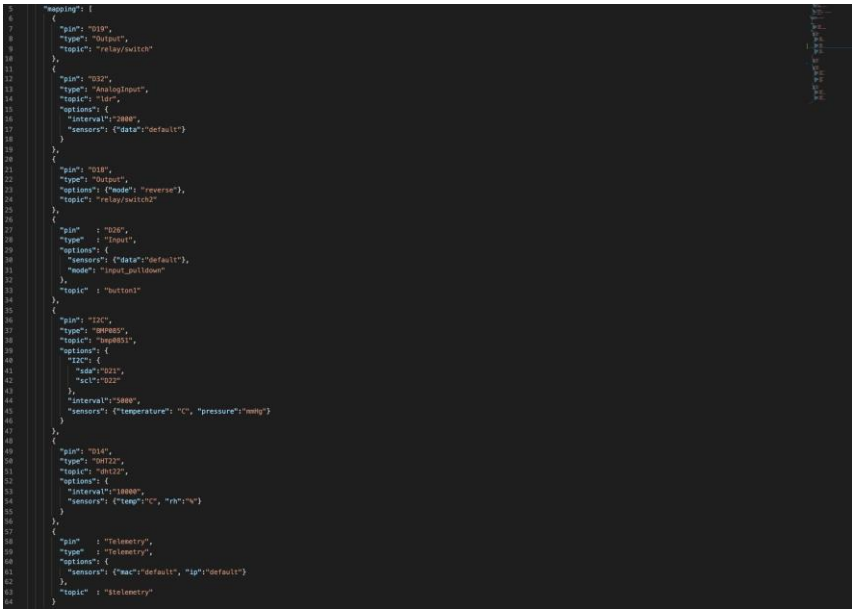
В об’єкті *mapping* виконується співвідношення модулів до конкретних сенсорі, опцій чи телеметрій с об’єкта в *deficeConfig*. Так у кожної ноди, сенсора, опції і телеметрії є унікальний в рамках девайсу ідентифікатор, який дозволяє сформувати повний топік до нього, що дає можливість передати його в модуль для передачі в нього даних. Таким чином масив mapping складається з об’єктів, що описують співвідношення модуля до топіка сутності. В кожного об’єкта є набір обов'язкових атрибутів:

pin — атрибут, що описує фізичний вихід на девайсі, з яким він спілкується з датчиком чи керованим пристроєм, type — атрибут, що описує тип модуля з бібліотеки тих, підтримка яких була реалізована

topic — атрибут, який описує співвідношення с сенсором, опція чи телеметрія конкретної ноди в форматі node-id/sensor-id

options — атрибут, в якому опціонально вказуються додаткові характеристики для роботи модуля, цей об’єкт відрізняється для кожного модуля і унікальний в рамках кожного.

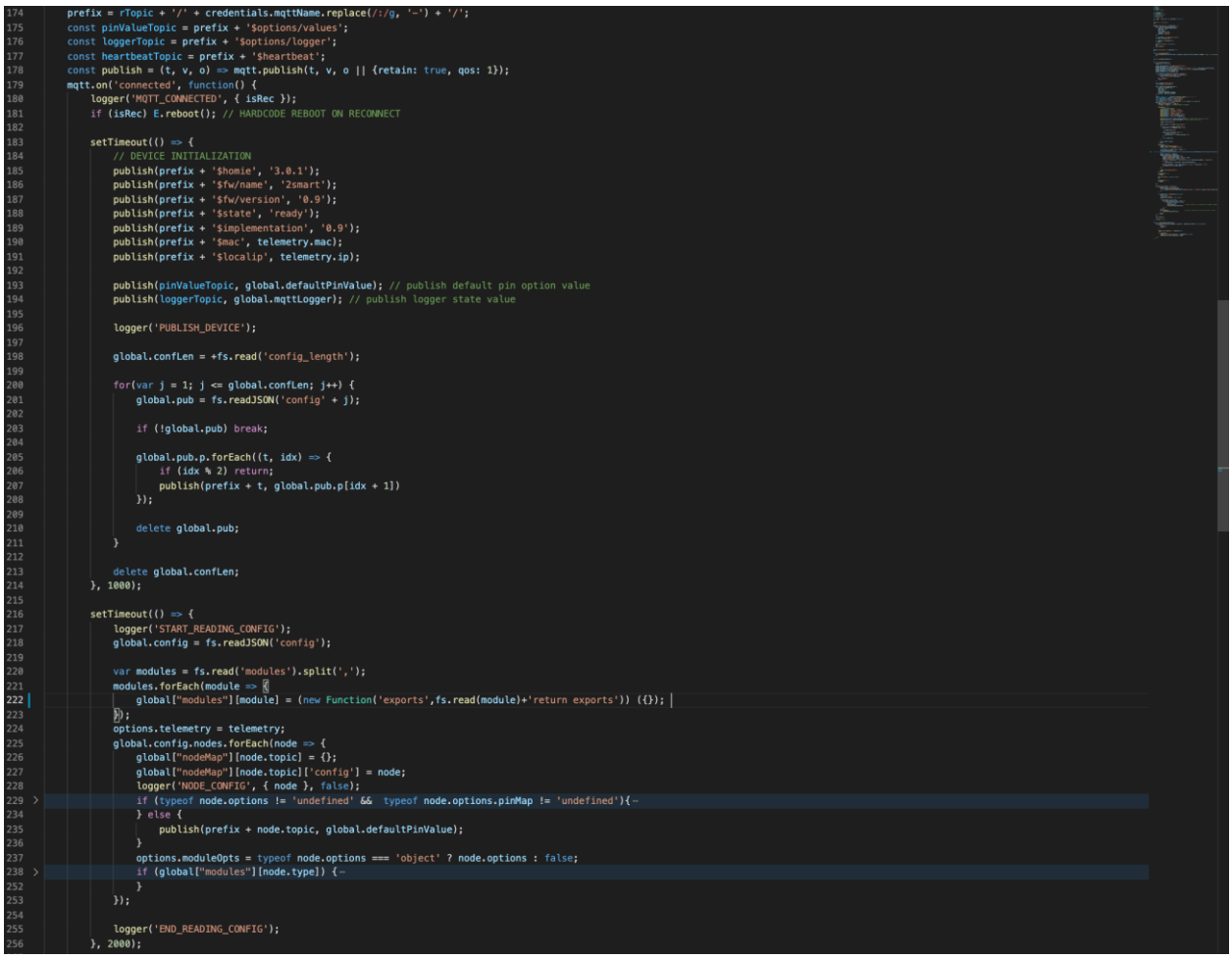
На рисунку (рис. 4.4) зображено приклад об’єкту mapping з варіацією декількох модулів



```
"mapping": [
  {
   "pin": "D19",
   "type": "Output",
   "topic": "relay/switch"
  },
  {
   "pin": "D32",
   "type": "AnalogInput",
   "topic": "ldr",
   "options": {
    "interval":"2000",
    "sensors": {"data":"default"}
   }
  },
  {
   "pin": "D18",
   "type": "Output",
   "options": {"mode": "reverse"},
   "topic": "relay/switch2"
  },
  {
   "pin"   : "D26",
   "type"  : "Input",
   "options": {
    "sensors": {"data":"default"},
    "mode": "input_pulldown"
   },
   "topic"  : "button1"
  },
  {
   "pin": "I2C",
   "type": "BMP085",
   "topic": "bmp085",
   "options": {
    "I2C": {
      "sda":"D21",
      "scl":"D22"
    },
    "interval":"5000",
    "sensors": {"temperature": "C", "pressure":"mmHg"}
   }
  },
  {
   "pin": "D14",
   "type": "DHT22",
   "topic": "dht22",
   "options": {
    "interval":"10000",
    "sensors": {"temp":"C", "rh":"%"}
   }
  },
  {
   "pin"   : "Telemetry",
   "type"  : "Telemetry",
   "options": {
    "sensors": {"mac":"default", "ip":"default"}
   },
   "topic"  : "$telemetry"
  }
```


Рис. 4.4 Приклад об’єкта з описом співвідношення модулів до сенсорів

Після формування структури конфігураційного файлу в основному виконуючому файлі було написана функція парсингу конфіга, яка по описаній конфігурації збирає необхідні дані з інших файлів і формує структуру топіків які потрібно відправити в брокер і підписатись.



```
prefix = rTopic + '/' + credentials.mqttName.replace(/:/g, '-') + '/';
const pinValueTopic = prefix + '$options/values';
const loggerTopic = prefix + '$options/logger';
const heartbeatTopic = prefix + '$heartbeat';
const publish = (t, v, o) => mqtt.publish(t, v, o || {retain: true, qos: 1});
mqtt.on('connected', function() {
    logger('MQTT_CONNECTED', { isRec });
    if (isRec) E.reboot(); // HARDCODE REBOOT ON RECONNECT

    setTimeout(() => {
        // DEVICE INITIALIZATION
        publish(prefix + '$homie', '3.0.1');
        publish(prefix + '$fw/name', '2smart');
        publish(prefix + '$fw/version', '0.9');
        publish(prefix + '$state', 'ready');
        publish(prefix + '$implementation', '0.9');
        publish(prefix + '$mac', telemetry.mac);
        publish(prefix + '$localip', telemetry.ip);

        publish(pinValueTopic, global.defaultPinValue); // publish default pin option value
        publish(loggerTopic, global.mqttLogger); // publish logger state value

        logger('PUBLISH_DEVICE');

        global.confLen = +fs.read('config_length');

        for(var j = 1; j <= global.confLen; j++) {
            global.pub = fs.readJSON('config' + j);

            if (!global.pub) break;

            global.pub.p.forEach((t, idx) => {
                if (idx % 2) return;
                publish(prefix + t, global.pub.p[idx + 1])
            });

            delete global.pub;
        }

        delete global.confLen;
    }, 1000);

    setTimeout(() => {
        logger('START_READING_CONFIG');
        global.config = fs.readJSON('config');

        var modules = fs.read('modules').split(',');
        modules.forEach(module => {
            global["modules"][module] = (new Function('exports',fs.read(module)+'return exports')) ({});
        });
        options.telemetry = telemetry;
        global.config.nodes.forEach(node => {
            global["nodeMap"][node.topic] = {};
            global["nodeMap"][node.topic]['config'] = node;
            logger('NODE_CONFIG', { node }, false);
            if (typeof node.options != 'undefined' &&  typeof node.options.pinMap != 'undefined'){-
            } else {
                publish(prefix + node.topic, global.defaultPinValue);
            }
            options.moduleOpts = typeof node.options === 'object' ? node.options : false;
            if (global["modules"][node.type]) {-
            }
        });

        logger('END_READING_CONFIG');
    }, 2000);
```

Рис. 4.5 Функція зчитування та обробки даних в файлі конфігурації

Даний файл конфігурації піддається мініфікації і завантажуються на вбудовану пам’ять пристрою і функція звертається до нього під час кожного завантаження пристрою. Крім цього функція відправляє в брокер необхідні по конвенції топіки з потрібними значеннями за замовчуванням (рис.4.5).

4.2 Створення модульної бази для роботи с сенсорами та керованими пристроями та розробка інструментів для програмного забезпечення

Для повноцінного функціонування програмного забезпечення необхідною складовою являється підключення необхідних модулів для роботи с фізичним датчиками та пристроями. Для зчитування даних та відправки команд на пристрої в розробці програмного забезпечення існує безліч підходів та кожний із зовнішніх пристроїв в основному потребує специфічного підходу. Для зменшення витрат пам'яті та ресурсів мікроконтроллера було запропоновано виносити бізнес логіку роботи с кожним типом пристрою в окремі модулі і підключати їх лише у випадку, коли вони потрібні, а отже описані в файлі конфігурації.

В ході розробки проекту було розроблено модулі для найпопулярніших стронніх типів датчиків та шин, а також закладено можливість створювати нові модулі самими користувачами чи розробниками.

Для коректної роботи з модулями файл з бізнес логікою має бути винесений в окрему папку, згідно документації, та названий таким чином, як буде підключатись в файлі конфігурації.

Output

Найпростішим прикладом являється модуля для роботи керованим виходами мікроконтролера. Яскравим прикладом являється включення діоду, приєднаного на потрібну ніжку мікроконтролера або ввімкнення вимкнення реле. Модуль був названий output, а бізнес логіка по роботі була занесена в файл *Output.js* (рис. 4.6).

Приклад роботи з output (цифровими пристроями) був взятий із стандартної бібліотеки Espruino. Модуль дозволяє відправляти на цифровий вихід керуючі команди 1 або 0, вмикаючи чи вимикаючи його. Стан цифрового пристрою відправляється в брокер в топіки с типом данных boolean, де 1 відповідає значенню true, а 0 — false відповідно.

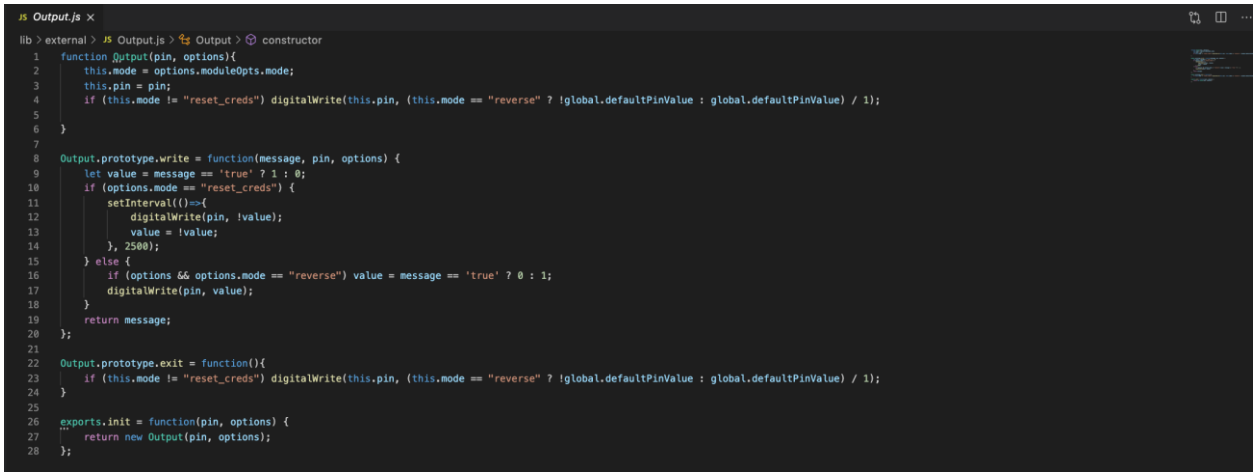


Рис. 4.6 Модуль Output

Модуль підтримує додатковий режим реверсу, що описується в розширених налаштування конфігураційного файлу і переводить режим роботи в інвертований, тобто значенню 1 відповідає false, а 0 — true.

Input

Наступним базовим модулем був Input. Це модуль який використовується для прийому сигналу з пристрою / датчика які працюють в режимі input / input_pullup / input_pulldown (отримання сигналу з пристрою на контролер).

Режими роботи модуля:

- *input_pull_X* — методи змінюють стан один раз при тривалому / нетривалому сигналі

- *input_pullup* — пристроїв працюють в інверсному режимі (Цифровий вхід з внутрішнім підтягуючим резистором ~ 40 кОм)

- *input_pulldown* — пристроїв працюють в звичайному режимі (Цифровий вхід з внутрішнім стягує резистором ~ 40 кОм)

- *input* - цифровий вхід для постійного слухання стану

На рисунку зображений код, що забезпечує роботи с сенсорами данного типу (рис.4.7).

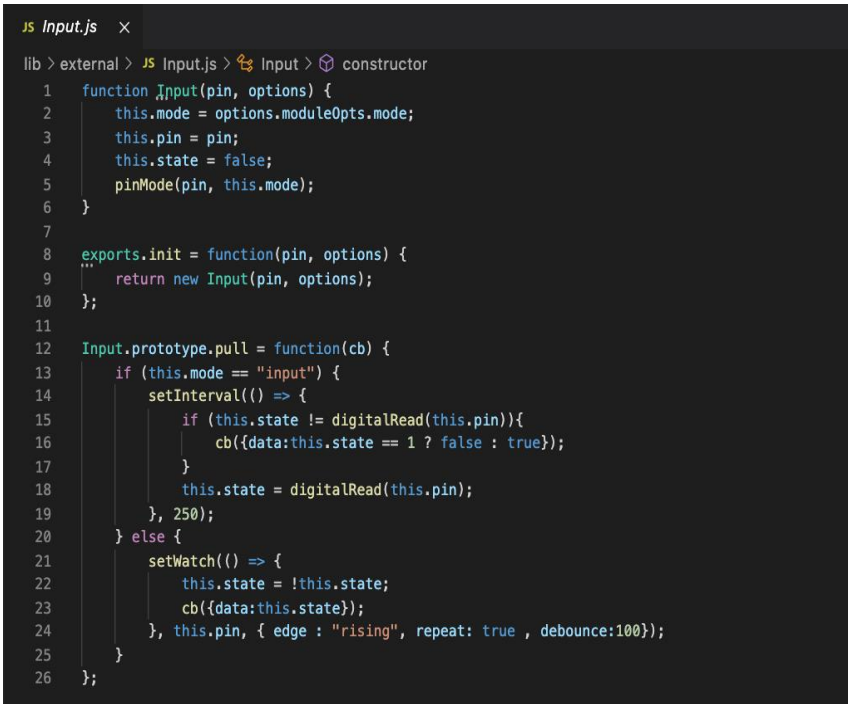


Рис. 4.7 Модуль Input

AnalogInput

Модуль який використовується для прийому сигналу з пристрою / датчика які працюють в режимі *input* і зчитують аналогові значення (отримання сигналу з пристрою на контролер)

У модулі AnalogInput реалізована логіка *reverse* яка дозволяє працювати з портом в инвертном режимі, для цього необхідно вказати *"options": {"mode": "reverse"}* (рис. 4.8).

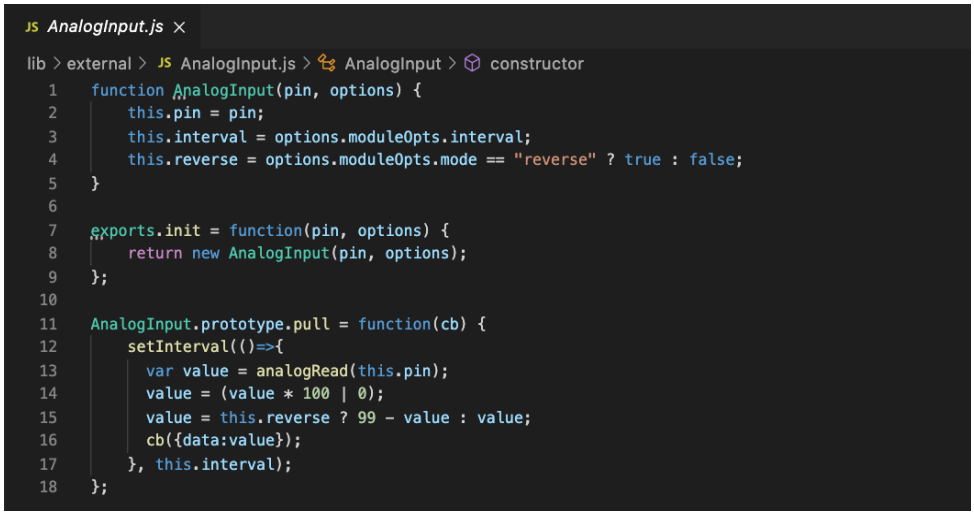


Рис. 4.8 Модуль AnalogInput

Telemetry

Додатковий стандартний модуль, який дозволяє передати в брокер мак адресу та айпі адресу пристрою, які він має в локальній мережі. Цей модуль створений для зручності користувача та швидкого пошуку парметрів пристрою.

Для налаштування модуля на певні порти в mapping необхідно поле type заповнити ім'ям модуля, а ключ поля sensors назвою полів об'єкту які повертається з модуля, тобто {"Mac": "default", "ip": "default"} а значення величиною одиниці виміру (рис. 4.9). Функція для отримання мак та айпі адрес являється стандартною в Espruino.

Рис. 4.9 Модуль Telemetry

DHT22

Далі модулі були реалізовані не так універсально, а специфічно під типи сенсорів. Так наприклад, модуль DHT22 — модуль який використовується для зчитування значень з DHT2x / AM230x / RHT0x датчиків температури і вологості.

Для налаштування модуля на певні порти в *mapping* необхідно поле *type* заповнити ім'ям модуля.

В поле *options.interval* вказати значення інтервалу зчитування інформації з DHT, значення задається в мілісекундах.

Доступні значення для температури та вологості:

- *"C"* - цельсій, *"K"* - кельвін.*"%"* - відсоткове представлення значення.

Як видно в даному модулі з'являється додаткова логіка по роботі с пристроєм (рис. 4.10).

DHT22.js

lib > external > DHT22.js > DHT22 > constructor

```
function DHT22(pin, options) {
  this.pin = pin;
  this.interval = options.moduleOpts.interval;
  this.sensors = {};
  for (sensor in options.moduleOpts.sensors){
    this.sensors[sensor] = {};
    switch(options.moduleOpts.sensors[sensor]) {
      case "K":
        this.sensors[sensor].divider = 1;
        this.sensors[sensor].offset = 273.15;
        break;
      case "C":
        this.sensors[sensor].divider = 1;
        break;
      case "%":
        this.sensors[sensor].divider = 1;
        break;
    }
  }
}

DHT22.prototype.read = function (cb, n) {
  if (!n) n=10;
  var d = "";
  var ht = this;
  digitalWrite(ht.pin, 0);
  pinMode(ht.pin,"output"); // force pin state to output
  // start watching for state change
  this.watch = setWatch(function(t) {
    d+=0|(t.time-t.lastTime>0.00005);
  }, ht.pin, {edge:'falling',repeat:true} );
  // raise pulse after 1ms
  setTimeout(function() {pinMode(ht.pin,'input_pullup');pinMode(ht.pin);},1);
  // stop looking after 50ms
  setTimeout(function() {
    if(ht.watch){ ht.watch = clearWatch(ht.watch); }
    var cks =
        parseInt(d.substr(2,8),2)+
        parseInt(d.substr(10,8),2)+
        parseInt(d.substr(18,8),2)+
        parseInt(d.substr(26,8),2);
    if (cks&&((cks&0xFF)==parseInt(d.substr(34,8),2))) {
      cb({
        rh : parseInt(d.substr(2,16),2)*0.1,
        temp : parseInt(d.substr(19,15),2)*0.2*(0.5-d[18]) + (ht.sensors.temp.offset || 0)
      });
    } else {
      if (n>1) setTimeout(function() {ht.read(cb,--n);},500);
      else cb({temp:"-", rh:"-"});
    }
  }, 50);
};

exports.init = function(pin, options) {
    return new DHT22(pin, options);
};

DHT22.prototype.pull = function(cb) {
    setInterval(()=>{
        this.read(cb);
    }, this.interval);
};
```

Рис. 4.10 Модуль DHT22

Додаткові модулі для роботи з різними типами сенсорів

По аналогії з іншими наведеними прикладами було створено бібліотеку з інших модулів, що дозволяють взаємодіяти з основними сенсорами та пристроями, а також закладено механізм розробки нових модулів експертними користувачами.

Перелік додаткових модулів та їх короткий опис:

1. BMP085 — модуль який використовується для зчитування значень температури і атмосферного тиску з цифрових барометрів BMP085 / BMP180 по засобу інтерфейсу I2C.

2. Register — модуль який використовується для роботи з утройство під контролем зсувного регістру

3. UltraSonicSensor — модуль який використовується для вимірювання відстані до перешкоди за допомогою ультразвукового сонара HC-SR04

4. MoveSensor — модуль який використовується для визначення руху в просторі за допомогою піроелектричного датчика руху HC-SR50

5. CCS811 — модуль для роботи з датчиком якості повітря в приміщеннях; представляє модуль цифрового газового датчика, який визначає широкий діапазон загальних летючих органічних сполук (TVOC), включаючи еквівалентні рівні діоксиду вуглецю (eCO 2) по засобу інтерфейсу I2C

6. DS18B20 — модуль який використовується для зчитування значень з датчика температури DS18B20

Після реалізації першого модулю було створено функцію, яка займається парсингом модулі, які треба підключати в бінарні файли і займається їх включенням в фінальну збірку. Далі код кожного із модулів зберігається на мікроконтролері і виконується, коли викликається з основної функції.

Розробка інструментів для завантаження та оновлення програмного забезпечення

Для встановлення програмного забезпечення на пристрій було створено ряд *.sh* скриптів, які запускають докер контейнер зі всіма необхідними інструментами для компіляції та завантаження програмного забезпечення на пристрій.

Перш за все користувачу необхідно ініціалізувати докер контейнер та встановити в ньому все необхідні інструменти. Для цього достатньо просто запустити скрипт *install_modules.sh* (рис. 4.11).

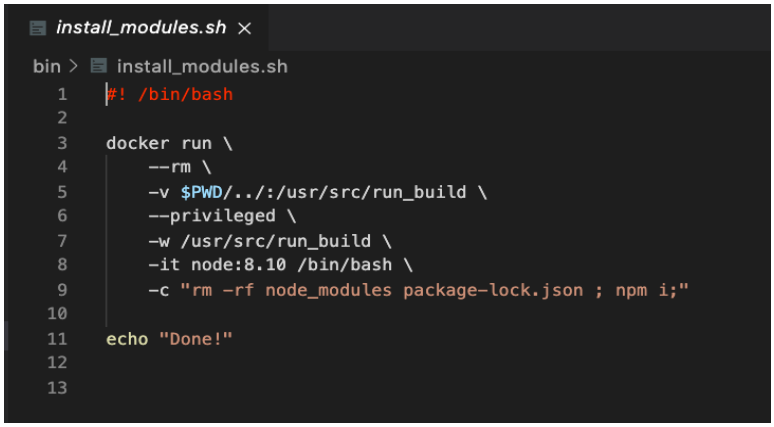


Рис. 4.11 Скрипт для ініціалізації середовища для завантаження ПЗ

Наступним кроком являється очищення мікроконтролера від старої прошивки та встановлення на нього середовище Espruino для можливості запуску розробленого ПЗ на ESP. Для цього необхідно просто запустити скрипт *flush.sh* (рис. 4.12), передавши першим аргументом порт, через який підключено контроллер до комп'ютера, а другим аргументом — назву файла конфігурації, що знаходяться в стандартній директорії.

Дану процедуру необхідно виконувати лише у випадку, коли на контролері не встановлена Esptuinoi, або встановлено старе програмне забезпечення. Під час виконання скрипту користувачу треба слідувати підказкам (наприклад перезавантажити вчасно девайс). Останнім кроком залишається збір та завантаження програмного забезпечення на

мікроконтролер. Для цього треба виконати скрипт *provision.sh*, передавши першим аргументом порт, через який підключено контроллер до комп'ютера, а другим аргументом — назву файла конфігурації, що знаходяться в стандартній директорії.

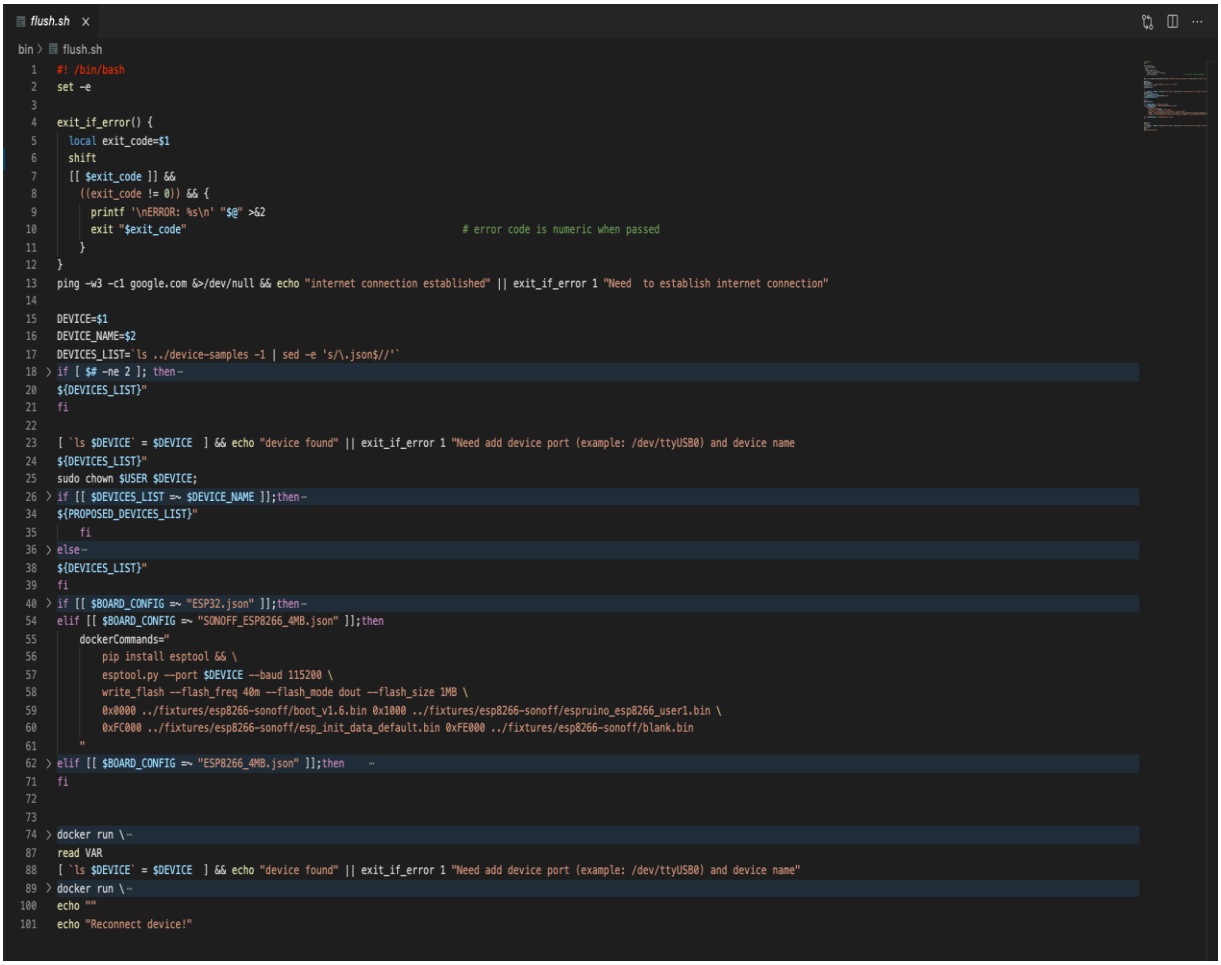


Рис. 4.12 Скрипт для встановлення Espruino

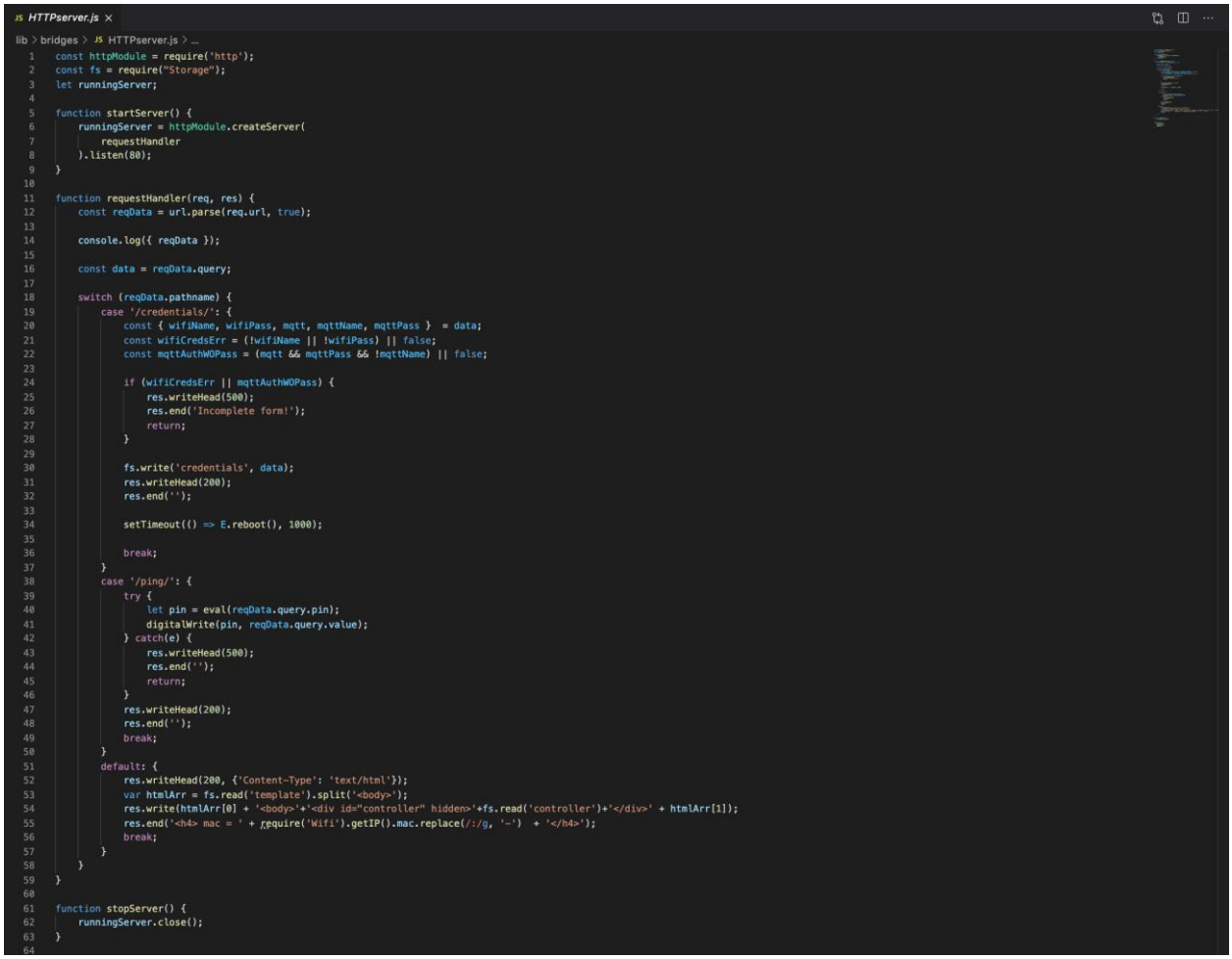
Кожного разу при запуску даного скрипта будуть збиратись необхідні бінарні файли і підтягуватись зміни в файлі конфігурації. Крім цього після першого програмування контролера і отримання ним айпі адреси з'являється можливість завантажити нову програмне забезпечення вже через технологію OTA, тобто без фізичного підключення до нього, а по протоколу TCP, знаючи лише його айпі адресу. Приклади запуску команди на перепрограмування та формат аргументів зображені на рисунку (рис. 4.13).

Рис. 4.13 Приклад запуску команди на перепрограмування

Таким чином було створено ряд скриптів для збору і завантаження програмного забезпечення на мікроконтролери. Важливим критерієм є те, що користувачеві достатньо встановити на комп'ютер лише систему Docker і прослідкувати інструкцію. Використання Docker дозволяю стандартизувати підхід до перепрограмування та не змушувати користувачів самостійно встановлювати на власну машину додаткові інструменти, всі вони будуть встановлені в рамках контейнеру докера після запуску відповідної команди. Крім цього важливою перевагою являється наявність можливості перепрограмування "по повітрю", що значно полегшує процес додавання нового функціоналу чи оновлення старого. Одним із недоліків являється те, що через технічні специфіки роботи Docker встановлення програмного забезпечення доступне лише с OS Linux.

4.3 Створення базового інтерфейсу для підключення пристроїв в локальну мережу

Після програмування пристрої розробленим програмним забезпечення в нього потрібно передати параметри підключення в локальну мережу та адресу на якому хості і порту знаходиться брокер. В першій версій програмного забезпечення дані параметри вкладались в файл конфігурації, але в процесі використання приладів було виявлено, що даний спосіб не є практичним і потребує постійного перепрограмування в разі зміни одного с параметрів. Для зручного використання пристроїв потрібно було створити постійний веб сервер для прийому REST запитів з необхідними параметрами. Першим кроком було програмування серверної частини, яка запускалась на мікроконтролері після його завантаження (рис. 4.14).



```
JS HTTPserver.js ×
lib > bridges > JS HTTPserver.js > ...
const httpModule = require('http');
const fs = require("Storage");
let runningServer;

function startServer() {
    runningServer = httpModule.createServer(
        requestHandler
    ).listen(80);
}

function requestHandler(req, res) {
    const reqData = url.parse(req.url, true);

    console.log({ reqData });

    const data = reqData.query;

    switch (reqData.pathname) {
        case '/credentials/': {
            const { wifiName, wifiPass, mqtt, mqttName, mqttPass }  = data;
            const wifiCredsErr = (!wifiName || !wifiPass) || false;
            const mqttAuthWOPass = (mqtt && mqttPass && !mqttName) || false;

            if (wifiCredsErr || mqttAuthWOPass) {
                res.writeHead(500);
                res.end('Incomplete form!');
                return;
            }

            fs.write('credentials', data);
            res.writeHead(200);
            res.end('');

            setTimeout(() => E.reboot(), 1000);

            break;
        }
        case '/ping/': {
            try {
                let pin = eval(reqData.query.pin);
                digitalWrite(pin, reqData.query.value);
            } catch(e) {
                res.writeHead(500);
                res.end('');
                return;
            }
            res.writeHead(200);
            res.end('');
            break;
        }
        default: {
            res.writeHead(200, {'Content-Type': 'text/html'});
            var htmlArr = fs.read('template').split('<body>');
            res.write(htmlArr[0] + '<body>'+'<div id="controller" hidden>'+fs.read('controller')+'</div>' + htmlArr[1]);
            res.end('<h4> mac = ' + require('Wifi').getIP().mac.replace(/:/g, '-')  + '</h4>');
            break;
        }
    }
}

function stopServer() {
    runningServer.close();
}
```

Рис. 4.14 Веб сервер для прийому запитів з параметрами підключення

Сервер приймає наступні необхідні параметри:

- назву локальної мережі
- пароль для підключення до локальної мережі
- адрес MQTT брокеру
- логін для підлючення до MQTT брокеру
- пароль для підлючення до MQTT брокеру

За замовчування пристрій перевіряє наявність параметрів на файлові системі і, якщо їх немає, переходить в режим роздачі точки доступу за замовчуванням. В такому випадку користувач може підключитись до цієї мережі і передати необхідні параметри за допомогою REST запиту. Після отримання параметрів, пристрій записує їх на файлову систему, автоматично

перезавантажується і після цього підключається до вказаної мережі та брокеру з отриманими параметрами.

Для очищення отриманих параметрів було створено окремий системний модуль для очистки парметрів з файлової системи після взаємодії з кнопкою, яка з'єднана з потрібною ніжкою мікроконтроллера.

Замінити параметри підключення пристрою до мережі можливо шляхом переводу девайса в режим точки доступу шляхом довгого замикання *GPIO0* (для ESP32 / ESP8266 це кнопка boot або flash).

Для цього необхідно підключити *моудль Output* в *extensions.modules*, потім вказати в *mapping* режим модуля *Output* в *reset_creds* (рис. 4.15).

Рис. 4.15 Приклад налаштування модуля зміни параметрів підключення

Таким чином зміну параметрів можна ініціювати як фізичним замиканням кнопки на ніжці, або зробити це через брокер, відправивши в потрібний топік значення true.Наступним кроком стала розробка базового веб інтерфейсу для передачі параметрів доступа. Для вирішення даної задачі було створено просту html сторінку з формою, що складається з необхідних полів для кожного з параметрів (рис 4.15).

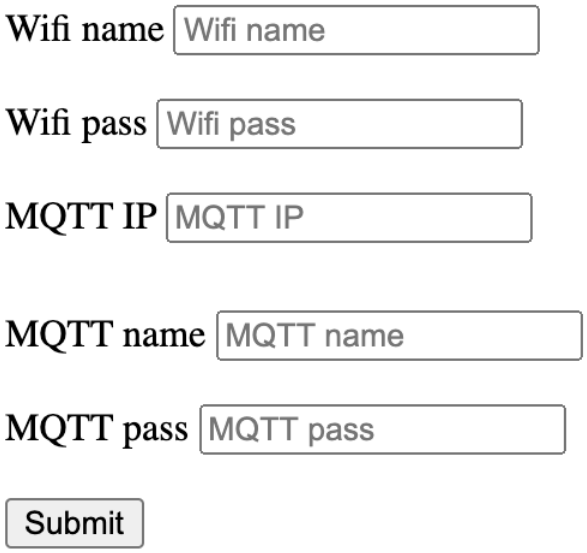


Рис. 4.16 Базова веб сторінка для передачі параметрів доступу

Таким чином за замовчування дана веб сторінка буде доступна на айпі адресі веб серверу 192.168.4.1, коли пристрій знаходиться в режимі точки доступу, а після підключення пристрою в локальну мережу буде можливість підключитись до неї по айпі адресі девайсу в локальній мережі.

4.4 Приклад тестування програмного забезпечення

Для тестування програмного забезпечення було використано стандартну реалізації на базі реалізації з відкритим кодом від компанії EMQX. MQTT брокер був налаштований на стандартному порту 1883 і хостится на домашньому персональному комп’ютеру. Таким чином пристрої з встановленим на них програмним забезпечення підключались до броокера по айпі адресі ПК та порту брокера.

Для розмежування параметрів доступу в інтерфейсі брокеру було налаштовано ACL (Access Control List), а саме базовий набір логінів та паролів, які давали доступ конкретному девайсу в конкретну виділену для нього область топіків в брокері.

Таким чином, знаючи параметри для підключення до локальній мережі, адресу брокера, а також логін та пароль для нього можна було передавати отримані дані через веб інтерфейс.

На рисунку (рис. 4.17) зображено стан девайсу в брокері після підключення. Для моніторингу стану девайсу використовувалась стандартна утиліта MQTT Explorer, яка по факту являється звичайним клієнтом пробера, але підключається до нього з правами адміністратора, що давало можливість бачити в керувати всіма девайсами, що підключені в нього.

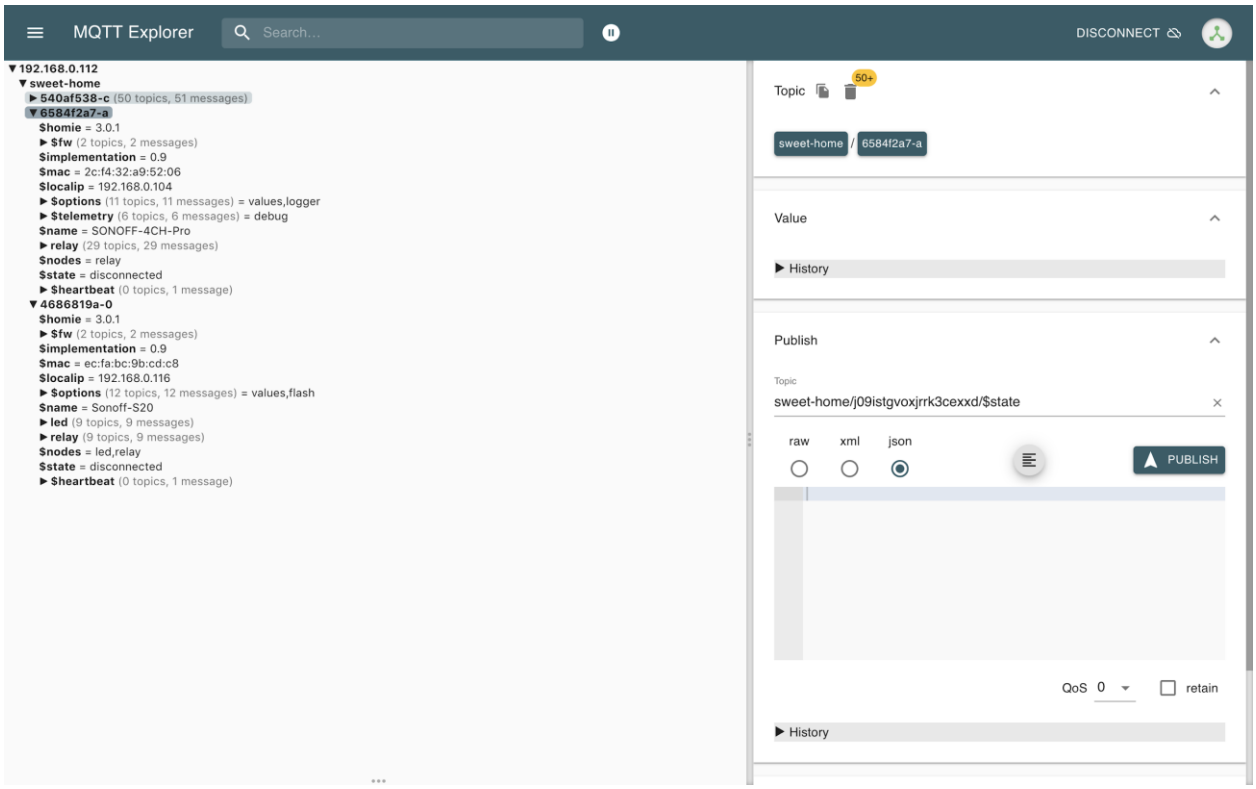


Рис. 4.17 Стан декількох девайсів в брокері

Для тестування всіх модульних компонентів було отримано ряд сенсорів та датчиків, та прототипи реальних пристроїв було зібрано на макетних платах по інструкція в відкритих джерелах.

Таким чином було протестовано наступні реальні пристрої та перевірено наступні сценарії:

- Включення лед підсвітки та діодів на макетних платах
- Отримання стану натиснення стандартних кнопок на макетній платі

- Зчитування показників з сенсорів температури, тиску, вологості
- Зчитування показників освітлення з фоторезисторів
- Керування декількома виходами через здвигові регістри
- Вимірювання відстані до перешкоди за допомогою датчиків
- Вимірювання показників якості повітря та летючих речовин після калібровки відповідного сенсору

Таким чином за допомогою макетної плати та звичайного мікроконтролера ESP32 було перевірено працездатність модулів, дані отримувались через брокер і їх можна було переглядати за допомогою утиліти MQTT Explorer.

Наступним кроком стало тестування програмного забезпечення на реальних промислових пристроях. Для цього було закуплено ряд девайсів, побудованих на основі мікроконтролерів ESP, які можна було комфортно демонтувати і встановити на них ПЗ.

Такими пристроями стали реле та розетки двох відомих брендів Sonoff та BlitzWolf:

- Sonoff S20 (однопортова розетка)
- Sonoff Basic (однопортове реле)
- Sonoff 4CH (чотирьох портове реле)
- Sonoff Dual (двохпортове реле)
- Blitz Relay (однопортове реле)
- Blitz SHP2 (однопортовий перехідник в розетку)

На цих 6 пристроях було перевірено програмне забезпечення в реальних користувальницьких сценаріях. До них було підключено різні побудові прибори (лампи, обігрівачі, пристрої контролю вологості та інше). Крім цього пристрій Sonoff Dual має на борту вихід для підключення аналогових датчиків температури, до нього було підключено один із сенсорів і значення температури заміялись в реальному часі.

Пристрої на макетних платах пройшли перевірку на функціонування, але не в тривалій часовій формі. А от пристрої від реальних виробників перевірялись часов та були протестовані в негативних сценаріях використання, а саме:

- коректно реагували на перебої з електроенергією
- вдало працювали і зберігали свій стан при перебоях в роботі локальної мережі, а після налаштування автоматично підключаються знову
- правильно обробляли обрив зв’язку з ПК та брокером

Для тестування було також написано невеликий ряд скриптів, які по налаштованому графіку вмикали чи вимикали навантаження на виході пристроїв. Такий сценарій повторює повністю реальні випадки і був використаний для ввімкнення освітлення для домашніх улюбленців.

Після тестування за допомогою стандартних утиліт та скриптів було проведено ряд тестів для перевірки можливості інтеграції зі сторонніми платформами. В ході аналізу існуючих платформ було обрано HomeBridge як одну із найцікавіших платформ. Завдяки ній за мінімальний срок було налаштовано голосове керування пристроями та можливість їз керування через веб інтерфейс платформи та мобільний додатком Home від iOS.

Для проведення даного експерименту платформу HomeBridge було розгорнуто на домашньому ПК по інструкціям з офіційного сайту. Платформа була розгорнута як окремий докер контейнер і приєдналась доброкеру, що був розгорнутий раніше.

Для налаштування було встановлено декілька плагінів платформи для роботи з брокером та двома голосовими ассистентами (Siri та Google) (рис. 4.18).

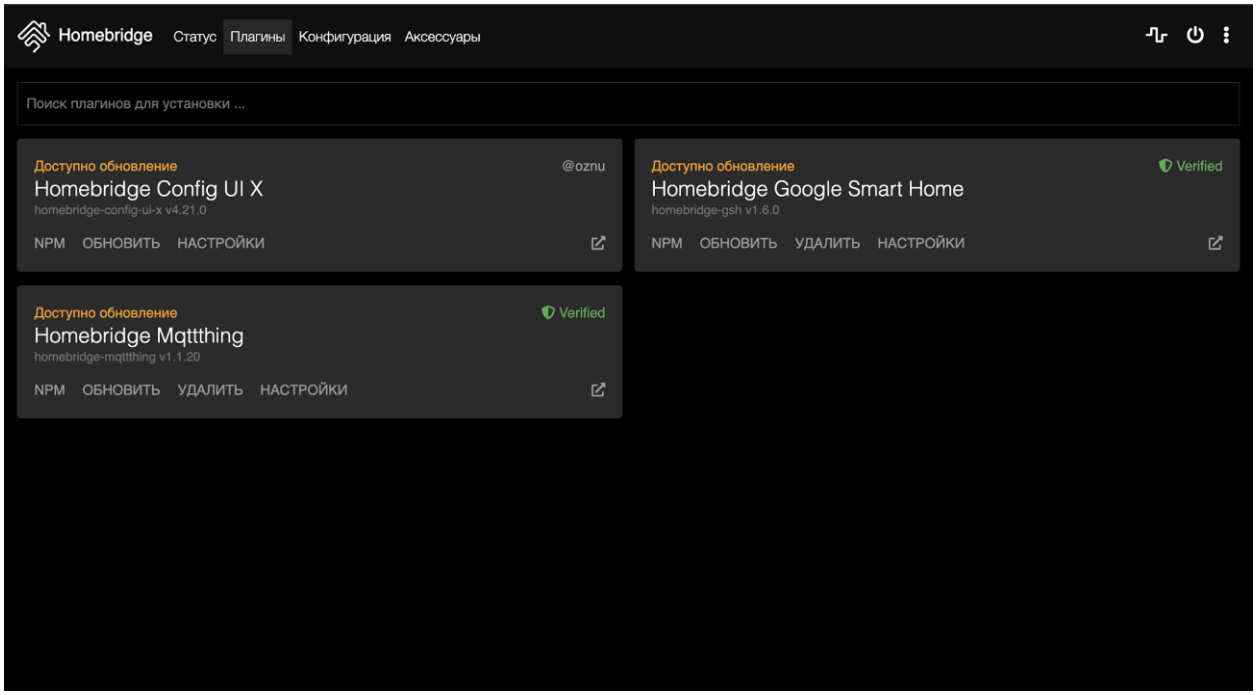


Рис. 4.18 Встановлення необхідних плагінів

Після налаштування плагінів, було описано конфігураційний файл для роботи з топіками реального пристрою. Цей процес дав зрозуміти, що архітектура дає можливість абстрагуватись від платформи і давати різним платформам працювати з абстракцією у вигляді топіків.

Файл конфігурації складається з опису аксесуарів, які далі будуть відображатись в веб інтерфейсі і взаємодіяти з голосовими асистенатами. В документації HomeBridge описані всі наявні аксесуари, а для тестування було обрано стандартний switch для перемикання навантаження (рис. 4.19).

Після збереження файлу конфігурації в веб інтерфейсі HomeBridge з’явився компонент для взаємодії з аксесуаром та відображення його стану та логів (рис. 4.20).

Після перевірки базового функціоналу наступним кроком стало налаштування голосових асистентів Siri та Google. Для цих цілей в Home Bridge є окремі плагіни, які були встановлені на попередньому етапі.

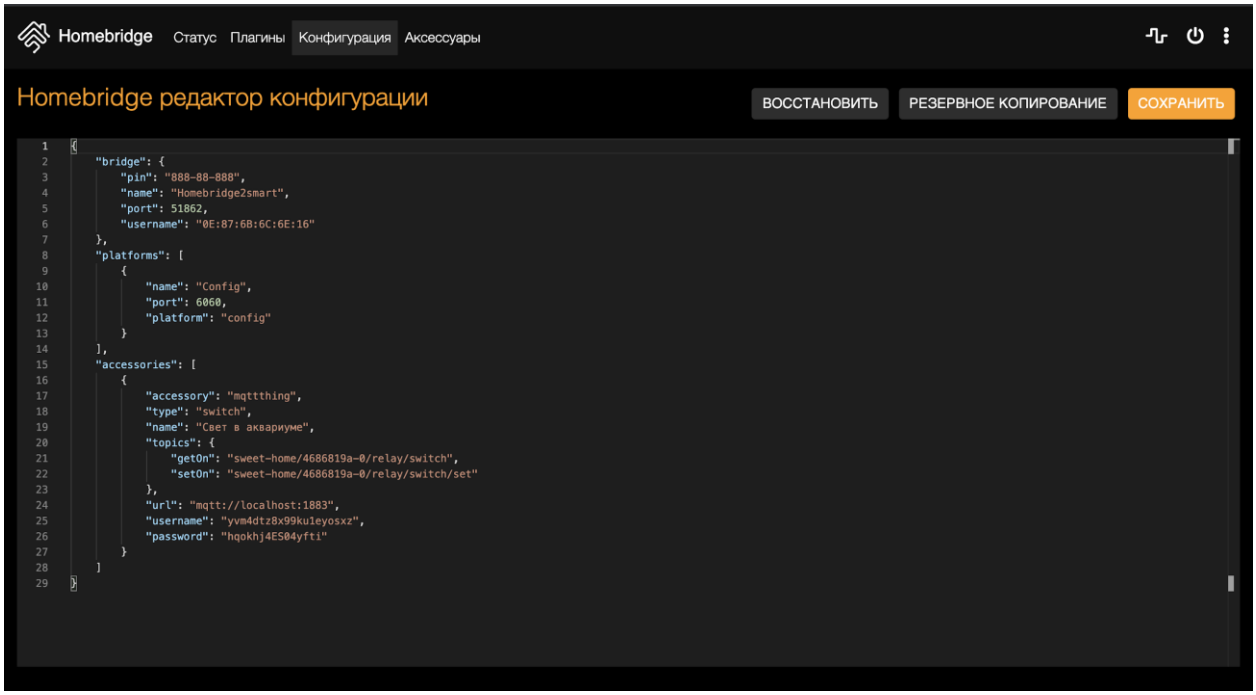


Рис. 4.19 Файл конфігурації в HomeBridge

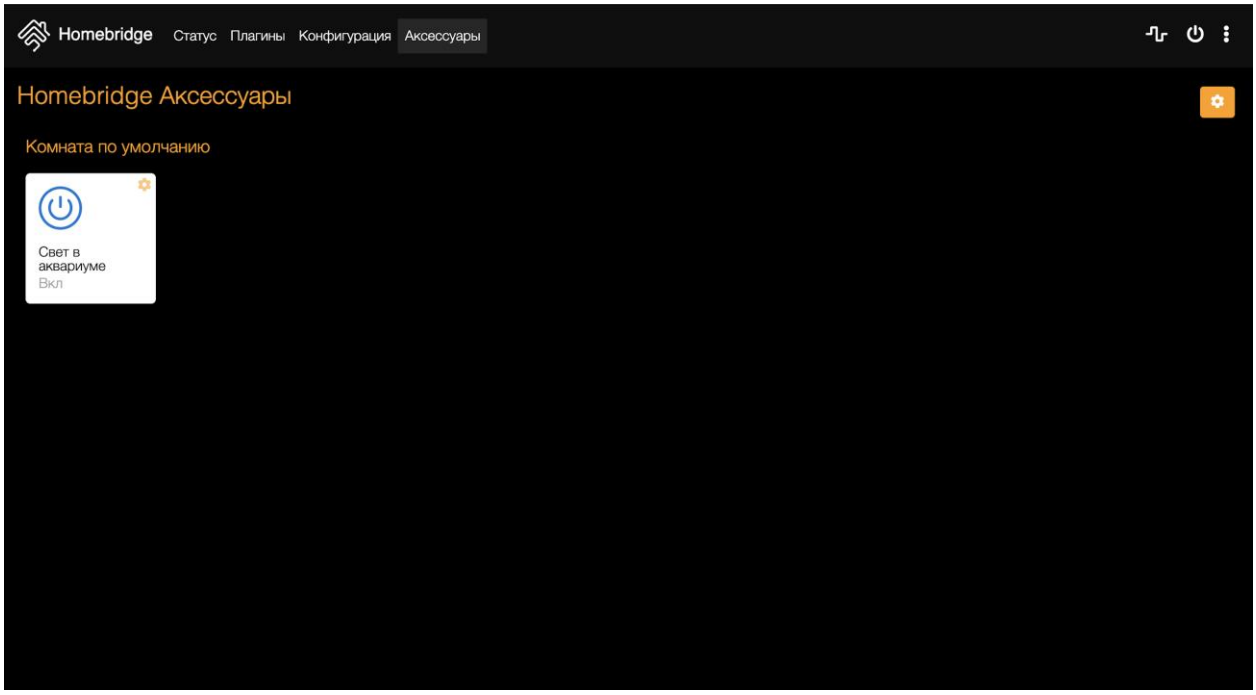


Рис. 4.20 Компонент для взаємодії з пристроєм

Для налаштування інтеграції з голосовими асистентами треба просто прослідкувати інструкції і встановити відповідні додатки на мобільному телефоні.

Після виконання необхідних дій, які займають не більше трьох хвилин пристрій, що керується розробленим програмним забезпеченням був доступний в додатку Home для iOS та Google Home. Саме це дало можливість вмикати та вимикати світло для домашніх улюбленців за допомогою голосових команд (рис. 4.21). По такому ж принципу можна додавати велику кількість аксесуарів, що доступні в HomeBridge, а саме:

- сенсори температуры
- сенсори руху
- сенсори освітлення
- датчики відкриття вікон та дверей
- сенсори якості повітря

Голосові асистенти коректно відпрацьовують з усіма стандартними аксесуарами і інтерактивно відповідають на поставлені питання.

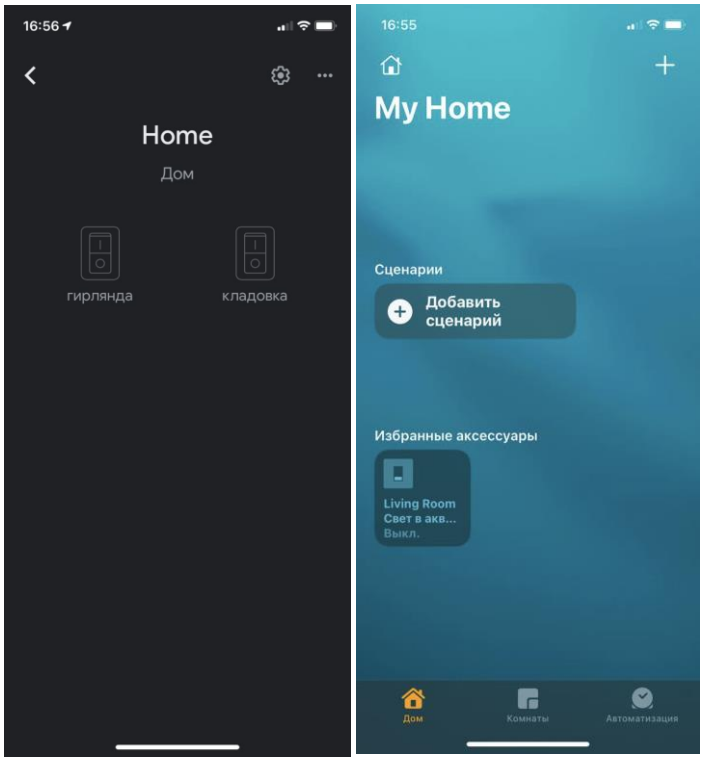


Рис. 4.21 Інтеграція з голосовими асистентами

Висновки

В четвертому розділі роботи було запропоновано власну реалізації програмного забезпечення пристроїв розумного будинку з удосконаленою архітектурою.

В процесі реалізації програмного забезпечення було виконано наступні задачі:

- створено бібліотеку із модулів для роботи з популярними датчиками та пристроями
- закладено механізм створення нових модулів сторонніми розробниками та користувачами
- створено CLI інструменті для встановлення необхідних інструментів для програмування, встановленовлення та оновлення програмного забезпечення на пристрої
- розроблено механізм завантаження програмного забезпечення по технології OTA
- реалізовано підтримку спеціалізованої IoT конвенції
- розроблено веб сервер та інтерфейс для підключення пристроїв в локальну мережу
- створено бібліотеку с конфігураційних файлів для тестування модулів в різних комбінаціях
- проведено тестування ПЗ на макетних пристроях
- проведено тестування в реальних умовах на промислових пристроях реальних виробників
- виконано інтеграції с платформою HomeBridge
- налаштовано голосове керування станом пристроїв за допомогою асистентів Siri та Google
- проведено стрес тестування та тестування негативних кейсів

На основі проведеного тестування було зроблено висновок про працездатність програмного забезпечення та відповідність його можливостей всім поставленим задачам.

Існуючі архітектури були покращені за рахунок:

- введення технології MQTT для абстрагування пристроїв від платформи
- розробки механізмів універсального підключення пристроїв до будь яких брокерів
- реалізації механізмів оновлення по повітрю
- реалізації можливості збору ПЗ на основі простого файлу конфігурацій
- створення бібліотеки модулів та можливості її розширення

РОЗДІЛ 5
РОЗРОБКА СТАРТАП-ПРОЕКТУ

В цьому розділі буде запропоновано реалізація стартап проекту на основі запропонованої удосконаленої архітектури для побудови пристроїв розумного будинку, проведено маркетинговий аналіз та обрано основні вектори розвитку проекту, а також оцінено його маркетингову цінність.

5.1 Опис ідеї проекту

Ідея проекту полягає у побудові пристроїв розумного будинку на базі удосконаленої архітектури, за рахунок використання реалізованого програмного забезпечення для пристроїв на базі мікроконтролерів ESP.

У таблиці 5.1 зображено зміст ідеї та можливі сфери для пошуку потенційних користувачі.

Таблиця 5.1. Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Використання розробленого програмного забезпечення в пристроях розумного будинку	Домашня автоматизація	Швидке та гнучке налаштування пристроїв багатьох типів
	Автоматизація промислових процесів	Стандартизація процесів управління пристроями в єдиній екосистемі

В ідеї стартап проекту пропонується використання розробленого програмного забезпечення в розумних пристроях. Головною ціллю проекта є пошук виробників розумних девайсів, або навіть звичайних побутових приладів з можливістю інтеграцію в них мікроконтролера ESP32 для запуску програмного забезпечення, яке дозволяє інтегрувати пристрій в системи керування або автоматизації.

Для валідації потенційних техніко-економічних переваг програмного забезпечення з популярними гравцями на ринку було проведено аналіз

існуючих пропозицій програмного забезпечення та проведено їх порівняльний аналіз з запропонованою ідеєю.

Для порівняння було обрано два популярних рішення, які були знайдені як альтернативні види запропонованого варіанту:

- ESPHome (Open Source проект, що ідеологічно схожий на запропонований варіант)
- Tasmota (програмне забезпечення для пристроїв компанії Sonoff)
- Blynk (екосистема для створення програмного забезпечення і мобільних додатків для керування)

Таблиця 5.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Характеристики проекту	(потенційні) товари/концепції конкурентів				W	N	S
		Мій проект	ESPHome	Tasmota	Blynk			
1.	Поріг входу	Високий	Низький	Середній	Високий			+
2.	Користувальницький досвід	Високий	Середній	Середній	Високий			+
3.	Стабільність роботи	Високий	Середній	Високий	Середній		+	
4.	Гнучкість	Високий	Високий	Низький	Високий		+	

В таблиці вказано оцінку кожного із конкурентів за шкалою Високий-Середній-Низький, де перший відповідає оцінці відмінно. В правій частині таблиці відмічено порівняльний аналіз представленого проект у порівняння з конкурентами. Аналіз конкурентів та вивчення їх позицій на ринку дало зрозуміти, що кожен має свою аудиторію, але повністю не відповідає всім потребам користувачів. В таблицю спеціально було винесено ті критерії, гарні і негативні показники в яких показали конкуренти.

5.2 Технологічний аудит ідеї проекту

В рамках даного підрозділу було проведено аудит технологічної можливості створення проекту.

Визначення технологічної здійсненності ідеї проекту передбачає аналіз складових які вказані в таблиці 5.3.

Таблиця 5.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність техно-логій
1.	Програмне забезпечення для пристроїв розумного будинку	Завантаження ПЗ	Наяна	Доступна
		Оновлення ПЗ	Наявна	Доступна
		Компіляція на основі файлу конфігурації	Наявна	Доступна
		Тестування	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: наявна та доступна на ринку				

В роботі було представлено варіант технічної реалізації проекту. Для повноцінного запуска в маркети продукт потребує доопрацювань, але вся технічна база наявна на ринку і проект можливий для реалізації.

5.3 Аналіз ринкових можливостей запуску стартап проекту

Для визначення актуальночті та доцільності розробки проекту потрібно провести аналіз ринкомих можливостей його запуску, тобто прооаналізувати всі переваги та загрозу, які можуть вплинути на маркетингову стратегію розвитку.

В таблиці наведено базову характеристику потенційного ринку, на який буде виходити проект, а саме ринок пристроїв розумних будинків. (таблиця 5.4).

Таблиця 5.4. Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн	10000 грн
3	Динаміка ринку	Зростаюча
4	Наявність обмежень для входу	Потреба сертифікація
5	Середня норма рентабельності в галузі (або по ринку), %	17%

Для аналізу потенціальних можливостей виходу на ринок було проаналізована декілька джерел, що свідчать про активний ріст самого ринку розумних пристрооїв, і найвіть наявність великої кількості ключових гравців не унеможливлює успіх проекта через значний попит з боку користувачів та виробників пристроїв.

Цільова аудиторія проекту це дві групи:

- виробники розумних девайсів, яким необхідне програмне забезпечення для функціонування їх апаратної частини
- DIY користувачі систем розумного дому, що самостійно створють пристрої

В таблиці наведено характеристику потенційних клієнтів та їх основні потреби, які можуть стати запорукою успіху стартап-проекту і мають обов’язково бути впроваджені(табл. 5.5).

Оскільки по аналізу конкурентів видно, що попит є з збоку двох груп користувачів, можна зробити висновок, що зацікавити першу групу буде не складно, а для зацікавлення другої треба буде контактувати на пряму і допомагати налаштовувати процеси, але основним фактором для них стане вартість проектів в перспективі за рахунок скорочення витратр [36].

Таблиця 5.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Гнучкість у створенні нових версій ПЗ	DIY користувачі	Поріг входу та вартість	Можливість самостійно реалізовувати нові модулі для сенсорів
2	Швидка розробка нових варіацій ПЗ	Виробники розумних пристроїв	Вартість проекту	Пришвидшений процес реалізації нових версій ПЗ

Для актуальної оцінки можливостей виходу на ринок було проаналізовано декілька факторів ризики, які можуть вплинути на успіх проекту(табл. 5.6).

Ризики існують, тож потрібно мати міцний фундамент у вигляді документів, сертифікатів, які підтверджують усі можливі наміри, результати тестувань та виділення основних переваг цього протоколу для більшої ефективності бездротових сенсорних мереж. Недивлячись на наявність рисків попит та роозвиток ринку свідчить про те, що у випадку гнучкогоо підходу і підлаштування і швидкої реакції на запити користувачів проект може отримати успіх. А за рахунок наявноості декількох цільових аудиторій вектори роозвитку можна змінювати або шукати нові, що значно підвищує шанс зайняти свою нішу.

Таблиця 5.6. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Попиту	Існуючі гравці можуть закривати потреби користувача	Аналіз та впроовадження інсструментів для вирішення інших задач
2.	Економічна	Зростання інфляції	Пошук можливостей для дешевшого тестування та розробки
№	Фактор	Зміст загрози	Можлива реакція компанії
3.	Конкуренція	Вихід нових гравців на ринок	Збільшення перевірок та гарних відгуків
4.	Науково-технічна	Швидкий розвиток науки	Моніторинг наукових новин та пошук нових шляхів вдосконалення проекту

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі(табл. 5.7).

Таблиця 5.7. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Tasmota,Sonoff, Blynk	Нові гравці	Ціноутворення	Задачі	Неякісні замінники
Висновки	Висока конкуренція свідчить про високий попит	Нові гравці можуть вийти з дозводім всіх існуючих конкурентів	Впливають на ціноутворення	Клієнти диктують основні умови на ринку	Можуть негартивно вплинути на авторитет

На основі проведеного аналізу можна сформувати перелік основних факторів, які свідчать про високу конкурентоспроможність проекту (табл. 5.8).

Таблиця 5.8. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Ціна інтеграція впливає на прийняття рішення. А наша ціна вигідніше, ніж у аналогів.
2	Актуальність	Ринок потребує вирішення поставлених задач
3	Попит	Розвиток ринку свідчить про активний попит
4	Гнучкість	Закладені механізми дозволять залишатись в ногу з потребами
5	Інноваційність	Робить українську науку на рівні с іншими країнами.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 5.10). Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення [37].

Таблиця 5.9. Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівняння з проектом						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	18		+					
2	Актуальність	17						+	
3	Попит	20						+	
4	Гнучкість	20	+						
5	Іноваційність	18						+	

З таблиць 5.9 та 5.10 що фактори конкурентноспроможності дозволять вийти на ринок, але успіху можна буде досягти лише за рахунок якісної реалізації та продуманої маркетингової політики.

Таблиця 5.10. SWOT- аналіз стартап-проекту

Сильні сторони: 1. Актуальність; 2. Інноваційність; 3. Вартість.	Слабкі сторони: 1. Відсутність довіри; 2. Велика витрата ресурсів до самих продажів на рекламу
Можливості: 1. Збільшення продаж; 2. Отримання державних замовлень на отримання послуг; 3. Розширення ринку за рахунок іноземних замовників; 4. Отримання тендерів на послуги.	Загрози: – Цінова конкуренція в зв’язку з появою нових гравців на ринку. – Різка зміна курсу гривні може привести до зменшення попиту, особливо з боку малих фірм.

Це знову підтверджує, що навіть незважаючи на свою специфіку, наш проект потребує значних зусиль для того, щоб увійти у ринок, зафіксуватися та пропонувати свої можливості своїм клієнтам(табл. 5.11).

На основі SWOT-аналізу розробляємо альтернативи ринкової.

Таблиця 5.11. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Стратегія нейтралізації ринкових загроз сильними сторонами стартапу	70%	3 місяці
2	Стратегія компенсації слабких сторін стартапу наявними ринковими можливостями	70%	3 місяці
3	Стратегія виходу з ринку	80%	6 місяців

З зазначених альтернатив обираємо стратегію компенсації слабких сторін стартапу наявними ринковими можливостями.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів(табл. 5.12).

Таблиця 5.12. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	DIY користувачі	Так	Високий	Середня	Середня
2	Промислові виробники	Так	Середній	Середня	Середня
Які цільові групи обрано: Під час аналізу потенційних груп споживачів було прийнято рішення що компанія буде працювати із обома групами користувачів і в майбутньому вибирати вектори подальшого роозвитку					

Для роботи з вибраним цільовими групами користувачів ринку необхідно сформувати базову стратегію розвитку(табл. 5.13).

Таблиця 5.13. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Підсилення сильних сторін стартапу за рахунок ринкових можливостей	Перемовини з виробниками і агресивний маркетинг для DIY корисстувачів	Виокремлення переваг цього способу у грошовому еквіваленті для майбутніх споживачів.	Стратегія підкріплення своїх переваг

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 5.14).

Таблиця 5.14. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Забирати існуючих	Так, недостатні функції по роботі з новими датчиками	Стратегія підкріплення своїх переваг і копіювання необхідних характеристик

На основі вимог споживачів з обраного сегменту до постачальника і продукту, а також в залежності від стратегії розвитку та стратегії конкурентної поведінки розробляємо стратегію позиціювання яка визначається у формування ринкової позиції, за яким споживачі мають ідентифікувати проект(табл. 5.15).

Таблиця 5.15. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Обширна документація та підримка у разі потреби	Гнучкість та швидка реакція на зміну	Експертиза та готовність підлаштовуватись під потреби клієнта	Якість. Термін служби. Гнучкість. Легкість. Дешивизна.

Результатом даного підрозділу є система рішень щодо ринкової поведінки компанії, вона визначає в якому напрямі буде працювати компанія на ринку задля успішного виходу і захоплення аудиторії.

5.5 Розроблення маркетингової програми стартап-проекту

Під час розроблення маркетингової програми першим кроком є розробка маркетингової концепції товару, який отримає споживач. У таблиці (табл. 5.16) підсумовуємо результати аналізу конкурентоспроможності товару.

Таблиця 5.16. Визначення ключових переваг концепції товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Конкурентоспроможності	Гнучкість і проостота	Найнижчий поріг входу і можливість інтеграції з будь якою платформою

Висновки

В результаті п’ятого розділу було представлено стартап-проект для просування запропонованої архітектури на ринок пристроїв розумного дому. В ході аналізу та розробки стартапу було проведено огляд та ознайомлення з основними гравцями в сфері та їх перевагами та недоліками. Результатом проекту являється продумана стратегія виходу на ринок, маркетинговий план, та обрані стратегії для роботи з потенційними клієнтами двох категорій. В результаті було зроблено висновок,що через високий попит

та активний розвиток сфери IoT попит в нових рішення є величезним і при наявності якісного продукту можливо отримати певний обсяг клієнтської уваги.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

Метою магістерської дисертації є вдосконалення сучасної архітектури побудови пристроїв розумного будинку. В результаті роботи було обрано нішу пристроїв на базі мікроконтролерів ESP, що займають більшу частину сучасних розумних девайсів на основі технології Wi-Fi і представлено власну реалізацію програмного забезпечення з вирішення проблем існуючих альтернатив.

В ході роботи було проведено аналіз сучасних технологій, що використовуються в основі пристроїв розумного будинко та вибрано оптимально для вирішення поставлених задач. Для повного розуміння потреб користувачів було опрацьовано більше двадцяти платформ, що дозволяють інтегрувати пристрої в одну екосистему і обрано стратегію для універсалізації підходу роботи з більшість платформ. Це було зроблено за рахуноок вибору MQTT протоколу в основі процесів переді даних між пристроєм та системою, саме за його рахунок було досягнуто абстракції від системи управління, а для стандартизації та самодекларативності девайсу було обрано спеціалізовану конвенцію.

Після вибору технохнологій та формування ідеї було опрацьовано можливе апаратне забезпечення для реалізації поставлених задач і обрано мікроконтролер ESP за основу пристроїв з удосконаленою архітектурою.

На основі отриманих даних в результаті аналізу було сплановано універсальну, гнучка та масштабовану архітектуру, яка дозволяє недосвідченим користувачам інтегрувати свої пристрої в ссистеми розумного дому будь-якого виробника за рахунок використання стандартних методів інтеграції на стороні платформ.

Для покращення користувальницького досвід в програмне забезпечення було впроваджено ряд інструментів для зручної компіляції та завантаження ПЗ на пристрій, а також розроблено механізм оновлення без фізичної взаємодії з девайсов.

Після реалізації кодової бази було проведено ряд тестових експериментів для перевірки працездатності продукту та йогоо можливостей. В результаті тестування було побудовано близько 20 пристроїв на макетних плата та перевірено можливість встановлення ПЗ на готові популярні пристрої від відомих виробників.

Для підтвердження вдосконалення архітектури розроблені пристрої було інтегровано з декількома системи контроль розумного дому та платформами автоматизації, а також проведено стрес тестування в реальних умовах використання.

Фінальною частиною роботи стала розробка стартап-проекту, в основі якого лежить використання представленого програмного забезпечення. В результаті було зроблено висновок, що проект може стати основою успішного проекту через виділені переваги та закладені механізми роботи.

На основі виконаної роботи можна зробити висновок, що в ході реалізації магістерської дисертації було сформовано ідею успішного проекту для розробки програмного забезпечення для великої кількості девайсів з можливість інтеграції в сучасні системи автоматизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smart home and appliances: State of the art / [Електронний ресурс] – Режим доступу:
https://publications.jrc.ec.europa.eu/repository/bitstream/JRC113988/kjna29750enn_1.pdf

2. Smart Home: Technologies with a standard battle / [Електронний ресурс] – Режим доступу:
https://ati.ec.europa.eu/sites/default/files/2020-06/Smart%20Home-%20Technologies%20with%20a%20standard%20battle%20%28v1%29.pdf

3. Applications, Systems and Methods in Smart Home Technology: A Review / [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/242630611_Applications_Systems_and_Methods_in_Smart_Home_Technology_A_Review

4. Smart Home: Architecture, Technologies and Systems / [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877050918305994

5. Github. Pointers for implementing the client / [Електронний ресурс] – Режим доступу: https://github.com/lucas-clemente/quic-go/.

6. J. Sissel. xdotool–fake keyboard/mouse input, window management, and more, 2013.

7. M. Chan, E. Campo, D. Estève, J.Y. Fourniols Smart homes - current features and future perspectives. Maturitas, 64 (2) (2009), p. 90

8. X. Fang, S. Misra, G. Xue, D. Yang Smart grid — the new and improved power grid: a survey. IEEE Communications Surveys & Tutorials,, 14 (4) , pp. 944-980 (2012)

9. Yang, C., Mistretta, E., Chaychian, S., & Siau, J. Smart home system network architecture (2017)

10. D.M. Han, J.H. Lim Design and implementation of smart home energy management systems based on zigbee.IEEE Transactions on Consumer Electronics,, 56 (3), pp. 1417-1425 (2010)

11. Qiao, X. M., Zhai, Y., Meng, P., Zhang, R. R., & Wang, C. . Research and application of intelligent interactive electricity technology based on fiber to the home. Electric Power Information & Communication Technology.(2013)

12. Kaneko, M., Arima, K., Murakami, T., Isshiki, M., & Sugimura, H. . Design and implementation of interactive control system for smart houses. IEEE International Conference on Consumer Electronics (pp.283-284). IEEE.(2017)

13. J. Palm Emergency management in the swedish electricity grid from a household perspective Journal of Contingencies & Crisis Management, 17 (1) , pp. 55-63 (2009)

14. Bueno, A. D. O. From Smart Cities to Social Cities:Technology to Support Community Life. CHI Conference Extended Abstracts on Human Factors in Computing Systems (pp.198-202). ACM.(2016).

15. Lin, L. I., Yao, G., & Tang, X. . Construction of interactive electricity service sytem for smart home of sino-singapore tianjin eco-city. Distribution & Utilization (2016).

16. C. Keles, A. Karabiber, M. Akcin, A. Kaygusuz, B.B. Alagoz, O. Gul A smart building power management concept: smart socket applications with dc distribution.

17. Alrawais, A., Alhothaily, A., Hu, C., & Cheng, X. . Fog computing for the internet of things: Security and privacy issues. IEEE Internet Computing, 21(2), 34-42 (2017).

18. Aminikhanghahi, S., Wang, T., & Cook, D. J. . Real-time change point detection with application to smart home time series data. IEEE Transactions on Knowledge and Data Engineering, 31(5), 1010-1023 (2018).

19. Anvari-Moghaddam, A., Monsef, H., & Rahimi-Kian, A. . Optimal smart home energy management considering energy saving and a comfortable lifestyle. IEEE Transactions on Smart Grid, 6(1), 324-332 (2014).

20. Chen, Y., & Kunz, T. Performance evaluation of IoT protocols under a constrained wireless access network. In 2016 IEEE International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT) (pp. 1-7) (2016, April).

21. Dabbagh, M., & Rayes, A. . Internet of things security and privacy. In Internet of Things From Hype to Reality (pp. 211- 238). Springer, Cham (2019).

22. Han, J., Choi, C. S., Park, W. K., Lee, I., & Kim, S. H. . Smart home energy management system including renewable energy based on ZigBee and PLC. In 2014 IEEE International Conference on Consumer Electronics (ICCE) (pp. 544-545) (2014, January).

23. Jacobsson, A., Boldt, M., & Carlsson, B. . A risk analysis of a smart home automation system. Future Generation Computer Systems, 56, 719-733 (2016).

24. Johnsen, F. T., Bloebaum, T. H., Avlesen, M., Spjelkavik, S., & Vik, B. . Evaluation of transport protocols for web services. In IEEE 2013 Military Communications and Information Systems Conference (pp. 1-6) (2013, October).

25. Kayal, P., & Perros, H. . A comparison of IoT application layer protocols through a smart parking implementation. 2017, IEEE 20th Conference on Innovations in Clouds, Internet and Networks (ICIN) (pp. 331-336) (2017, March).

26. Komninos, N., Philippou, E., & Pitsillides, A. . Survey in smart grid and smart home security: Issues, challenges and countermeasures. IEEE Communications Surveys & Tutorials, 16(4), 1933-1954 (2014).

27. Kumar, S. Ubiquitous smart home system using android application. arXiv preprint arXiv:1402.2114 (2014).

28. Lee, S., Kim, H., Hong, D. K., & Ju, H. . Correlation analysis of MQTT loss and delay according to QoS level. In 2013 IEEE The International Conference on Information Networking (ICOIN) (pp. 714-717) (2013, January).

29. De Caro N, Colitti W, Steenhaut K, Mangino G, Reali G Comparison of Two Lightweight Protocols for Smartphone-based Sensing IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT) Namur pp. 1-6 (2013).

30. Vergara E J, Prihodko M, Nadjm-Tehrani S Mobile Location Sharing: An Energy Consumption Study e-Energy pp 289-290 (2013)

31. Colitti W, Steenhaut K, De Caro N Integrating Wireless Sensor Networks with the Web Proc. IP+SN Chicago, USA (2011)

32. Lampkin V et al. Building smarter planet solutions with MQTT and IBM WebSphere MQ telemetry IBM, ITSO (2012)

33. Suresh P, Daniel J V, Aswathy R H A state of the art review on the Internet of Things (IoT) History, Technology and fields of deployment (2014)

34. J. M. Stewart. “Vehicle location and position monitoring system using satellite navigation and cellular telephone.” Vehicle Location and Fleet Management Systems, IEE Colloquium on. IET, (1993).

35. H. Chaouchi. “The internet of things: connecting objects. John Wiley & Sons” (2013 Feb).

36. O. Vermesan and P. Friess (Eds.). Digitizing the Industry - Internet of Things Connecting the Physical, Digital and Virtual Worlds, ISBN: 978-87-93379-82-4, River Publishers, Gistrup, (2016)

37. Mulas, Victor; Anastasia Nedayvoda, and Ghia Zaatari. Creative Community Spaces. Spaces that are Transforming Cities into Innovation Hubs.(2017)